FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Full Body Interaction in Serious Games for Rehabilitation

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor/in der technischen Wissenschaften

by

Christian Schönauer
Registration Number 9927275

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Mag. Dr. Hannes Kaufmann

The dissertation has been reviewed by:

(Priv.-Doz. Hannes Kaufmann) (Prof. Albert Rizzo)

Wien, 24.08.2015

(Christian Schönauer)

Erklärung zur Verfassung der Arbeit

Christian Schönauer
Am Tabor 12/2/9, 1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to thank everybody without whom this PhD thesis would not have been possible: First of all, I want to thank my advisor Priv.-Doz Hannes Kaufmann for giving me the opportunity to join his research group, his support and feedback and for countless inspiring discussions throughout the years. I am very grateful for his guidance as well as for giving me the freedom to follow my own ideas. I would also like to thank Prof. Christian Breiteneder for his knowledgeable advice and feedback concerning various topics including this thesis.

Furthermore, I would like to extend my thanks to the members of the VR Group at Vienna University of Technology for their valuable support and feedback. In particular, I would like to thank Dr. Annette Mossel, Georg Gerstweiler, Emanuel Vonach, Michael Bressler, David Zeller and Mathis Cisinko for their contributions to the framework described in Chapter 6. Furthermore, I want to thank Thomas Pintaric, Ferdinand Pilz and Markus Holzer for their contributions to the implementation of the motion capture system described in Chapter 4. I also want to thank Dr. Peter Kán, Khrystyna Vasylevska, Iana Podkosova and Peter Fikar for the good collaboration, valuable discussions and feedback. I would like to extend my thanks to Cris Voldum, Jens Juul Jacobsen, Jeppe H. Nielsen and Jørgen Krabbe from Serious Games Interactive for their contributions to the implementation of the game described in Chapter 7. Furthermore, I want to thank Dr. Stephanie Jansen-Kosterink, Prof. Miriam Vollenbroek, Dr. Rianne Huis in't Veld and Leendert Schaake from Roessingh Research and Development for their cooperation in the evaluation of the game. This work was partially funded by the European Union within the PLAYMANCER project (FP7-ICT-215839). Part of the work presented in this thesis has been conducted at MIT Media Lab. I would like to thank Prof. Ramesh Raskar for giving me this great opportunity. Furthermore, I want to thank Dr. Kenichiro Fukushi and Dr. Alex Olwal for their support and many useful discussions.

I would like to thank my parents for their support and encouragement to pursue my ideas. Furthermore, I want to thank my brother Robert and my friends, especially Clemens and Dominik for manifold support and helping me see my work from a distance.

Finally, I would like to thank my wonderful partner Christine for her endless encouragement and support.

Abstract

Serious games and especially their employment in healthcare applications are an active and rapidly growing area of research. Video games are expected to increase patient participation through motivational environments and to provide feedback, when repetitive rehabilitation exercises can be incorporated into game interactions. However, wide-spread use is hindered by several challenges regarding the availability and costs of input technologies, the workflow in an every-day clinical environment, the effort of application development and proof of efficacy. This thesis contributes to a solution to these problems in multiple dimensions.

An affordable flexible full body Motion Capture (MoCap) system has been developed, providing methods to generate a customized skeleton model for a user and fit it to optical tracking data in real-time. The MoCap data can be used as input to a serious game and to guide the patient in his relearning process (e.g. correcting errors in movement patterns), which is done by a therapist during conventional occupational or physical therapy. Furthermore, the algorithms were integrated in a workflow, which can be handled in an every-day clinical environment.

In addition, a low-cost MoCap system has been developed based on RGB-D sensors and evaluated as an alternative input modality for a home-based or telerehabilitation scenario.

However, muscle activity not always results in visible motions and might be difficult to track using conventional MoCap devices and therefore biosignal acquisition systems are also used for input.

Developing applications and serious games for rehabilitation, especially with a Virtual Reality (VR) setup, requires a lot of time and effort, because in addition to the implementation of game logic and content, often input/output devices, such as the above, have to be integrated and their data processed. Therefore, for serious games in rehabilitation and other VR applications in research and teaching we have developed a powerful framework - ARTiFICe -, that is lightweight and flexible and easily integrates new devices and technologies. Furthermore, it incorporates modules for interaction, distribution in multi-user scenarios and haptic feedback.

Finally, a serious game has been designed and implemented based on the ARTiFICe framework targeting rehabilitation of patients with musculoskeletal chronic pain of the lower back and neck, a group that has previously been neglected by serious games. The game was evaluated in a user study with a sample of ten adults with musculoskeletal pain and showed potential efficacy, clearly motivating patients to perform their exercises.

Kurzfassung

Serious Games und besonders ihr Einsatz in medizinischen Anwendungen sind ein sehr aktives und schnell wachsendes Forschungsfeld. Es ist zu erwarten, dass Computerspiele die Partizipation von PatientInnen durch motivierende Umgebungen verbessern und Feedback bieten, während repetitive Rehabilitationsübungen in Spielinteraktionen verpackt werden. Die weite Verbreitung wird derzeit allerdings durch verschiedene Herausforderungen behindert. Diese betreffen vor allem die Kosten und die Verfügbarkeit geeigneter Eingabetechnologien, den Workflow im klinischen Alltag, den Aufwand der Spieleentwicklung und den Nachweis klinischer Veränderungen.

In dieser Arbeit wurde ein erschwingliches flexibles Motion Capture- (MoCap) System entwickelt, welches Methoden zur Verfügung stellt, um automatisch ein Skelettmodel für BenutzerInnen zu erstellen und dieses in Echtzeit an optische Trackingdaten anzupassen. Die resultierenden MoCap-Daten können zur Steuerung eines Serious Games verwendet werden und PatientInnen in ihrem Lernprozess anleiten (z.B.: Korrektur von Fehlern in Bewegungsmustern). Diese Rolle wird während konventioneller Physio- oder Ergotherapie üblicherweise von TherapeutInnen übernommen. Weiters wurden die Algorithmen in einen Workflow integriert, der dem klinischen Alltag entspricht.

Zusätzlich wurde ein kostengünstiges MoCap-System basierend auf RGB-D Sensoren entwickelt und als Alternative für ein heimbasiertes oder Telerehabiliations-Szenario evaluiert.

Muskelaktivität resultiert allerdings nicht immer in sichtbaren Bewegungen, weshalb sie oft schwer mit konventionellen MoCap-Systemen zu erfassen ist. Deshalb wurden Biosignalverstärker ebenfalls als Eingabegeräte eingesetzt und getestet.

Die Entwicklung von Anwendungen und Serious Games für die Rehabilitation ist zeitintensiv und aufwändig, speziell mit einem Virtual Reality (VR) System. Oft müssen zusätzlich zu Spiellogik und Inhalten auch Ein- und Ausgabegeräte, wie die oben angeführten, integriert und ihre Daten verarbeitet werden. Aus diesem Grund wurde für Serious Games in der Rehabilitation und andere VR-Anwendungen in Forschung und Lehre ein umfassendes und flexibles Framework - ARTiFICe - entwickelt, das es erlaubt neue Geräte und Technologien einfach und transparent zu integrieren. Des Weiteren stellt es Module für die Interaktion, die Verteilung von Daten in Multi-User Szenarien sowie für haptisches Feedback zur Verfügung.

Schließlich wurde ein Serious Game entworfen und implementiert, das auf dem ARTiFICe Framework basiert und die Rehabilitation von PatientInnen mit chronischen Nacken- und Rückenschmerzen zum Ziel hat, weil diese Gruppe bisher in der Entwicklung von Serious Games keine Beachtung gefunden hat. Das Spiel wurde in einer Benutzerstudie mit einer Gruppe von zehn PatientInnen evaluiert. Dabei motivierte es die StudienteilnehmerInnen ihre Rehabilitationsübungen auszuführen und zeigte Potential klinische Veränderungen betreffend.

Contents

CHAPTER 1

Introduction

1.1 Motivation & Problem Statement

Rehabilitation is an important factor in the remobilization of people following accidents, injuries, stroke and other indications. The success of rehabilitation is on the one hand important for the individual (self-worth, independent living etc.), on the other hand rehabilitation is a major cost factor of today's health care and social systems. Therefore, efficient treatment can help lower costs for sick leave and care-giving expenses. The use of technologies such as Virtual Reality (VR) environments or biofeedback has been shown to be beneficial for the treatment progress [80, 57, 157] and could very well help to increase therapy efficacy on a wider scale in the future.

Many health care professionals consider motivation of the patient an important determinant of successful rehabilitation [87]. Firstly, during rehabilitation a patient usually follows an exercise program supervised by a therapist that requires the patient's active engagement. Secondly, to increase the intensity of this exercise program or to extend the inpatient rehabilitation, the therapist can assign home exercises. These home based exercise programs have shown to be effective, e.g. for chronic pain rehabilitation [84], but low adherence to these programs is problematic [142].

Using serious games, a motivational environment can be created and it is hypothesized patient participation in in-patient therapy as-well-as adherence to home based exercise programs can be increased. A virtual game environment can provide both distraction and incentives (game play rewards) to the patient and therefore increase motivation to engage in exercises [112].

Besides motivation, feedback and repetition are the other two key concepts shared among rehabilitation applications, which make serious games a useful addition to conventional therapy [112]. Thus repetitive rehabilitation exercises can be incorporated into game interactions, while the game system can provide feedback on the user's performance.

Especially for a home based (telerehabilitation) scenario it is crucial for a technical system to provide the patient with qualified feedback of performance [79, 80] and/or measure progress so a (remote) therapist can give feedback on the results later (e.g. via a web portal as described

in [20]). Therefore, the setup has to monitor patients' movements during exercising. By incorporating appropriate input and technologies for full body interaction and output modalities [62, 83], an application can monitor and guide a patient in the relearning process and give feedback on movement patterns. The goals of therapy for which serious games are applied are very diverse, including upper limb rehabilitation [23], balance restoration [12] or rehabilitation of specific body-parts, e.g. wrist [34], or training functional activities of everyday life in occupational therapy. A wide variety of hardware and software setups have been built in the past for these purposes, but most of the present systems are custom-tailored to a specific purpose. In this context, a general flexible solution applicable for various rehabilitation scenarios could save resources and should include a motion capture system capable of tracking arbitrary full body movements and integration with appropriate output devices. We have created such a system, that in addition is affordable, and provides a workflow that can be handled in a therapy scenario. Aspects of our work on this system have been published in [129, 130, 125] and are now consistently described in Chapter 4 of this thesis. Furthermore, feedback on a number of physiological measurements e.g. muscle activity [157], has been shown to be useful in a rehabilitation scenario. Therefore, we have integrated biosignal acquisition devices in our system as described in Chapter 6.

Developing applications and serious games for rehabilitation requires a lot of time and effort, because in addition to the implementation of game logic and content, input/output devices and technologies have to be integrated. Support for integration of non-desktop devices is usually not available in standard game engines. With a suitable framework on the other hand the development process can be significantly simplified and sped up. Ideally, such a framework should be flexible and extensible to easily integrate new devices and technologies and incorporate an up to date game engine and editor to support development of serious games for rehabilitation. Currently available frameworks either lack flexibility, a comprehensive development environment and/or generate high licensing cost. Therefore, we have developed our ARTiFICe framework, which provides an affordable solution to these issues. It supports Virtual Reality (VR) and Augmented Reality (AR) scenarios and is introduced in [93] and detailed in Chapter 6 of this thesis.

Applications in the medical domain and more specifically rehabilitation pose certain challenges on the system design as a whole, the tracking technologies as well as game design and workflow. These challenges make it hard to adapt conventional exergames or gaming systems intended for the consumer market for rehabilitation purposes. However, in some contexts consumer devices can offer a low-cost alternative to specialized equipment. Chapters 5 and 6 describe how a consumer MoCap device can be used to aid rehabilitation and how it is integrated in our framework respectively. Furthermore, Chapter 7 provides game design and workflow considerations in a concrete rehabilitation scenario.

Serious games in motor-rehabilitation have become a very active field of research within the last couple of years. They have been applied successfully with multiple indications, such as stroke, traumatic brain- and spinal cord and other injuries. However, an indication that has been neglected so far is the rehabilitation of patients suffering from chronic musculoskeletal pain. We have developed a serious game targeting this specific purpose as detailed in Chapter 7.

Rehabilitation for chronic pain as well as for many other indications is a multidisciplinary approach such as cognitive behavioral therapy containing psychological or social aspects. How-

ever, this thesis concentrates on the technical aspects providing tools to support rehabilitation that focus on human (full body) movements, i.e. motor rehabilitation and physical activation.

1.2 Contribution

The contribution of the work presented in this thesis is summarized in the following list:

- Development of our own marker-based full body motion capture system (Chapter 4):
  - Methods and algorithms for creation of a skeleton model and tracking from marker positions, that have not been published for commercial MoCap systems.
  - Adaptation and integration of the used algorithms in a workflow suitable for the requirements of serious games in rehabilitation, especially in the field of chronic pain rehabilitation.
  - Flexibility and accuracy at an affordable price.
- Design and development of a low-cost MoCap solution with RGB-D sensors (Chapter 5):
  - A flexible networked software environment.
  - Almost arbitrary scaling of the capture area.
- Design and development of a VR/AR/multimodal feedback framework with an emphasis on the requirements of rehabilitation applications (Chapter 6):
  - A complete, flexible middleware/tracking framework for various VR/AR setups
  - Modules for interaction distribution and visualization
  - Integration of the two MoCap solutions described above
  - Integration of new, low-cost controllers like Microsoft Kinect, Razer Hydra and 3D Connexion SpaceNavigator.
  - Integration of biosignal acquisition devices for the assessment of physiological measurements, especially muscle activity.
  - Integration of vibrotactile actuators for haptic feedback.
- Design, implementation and evaluation of a serious game targeting rehabilitation of chronic pain (Chapter 7):
  - A game customized to the medical requirements of patients suffering from chronic pain of the lower back and neck.
  - Evaluation with ten patients suffering from chronic pain.

1.3 Resulting Publications

The work presented in this thesis has appeared in the following peer reviewed publications and technical reports:

1.3.1 Peer Reviewed

[1] Peter Fikar, Christian Schönauer, and Hannes Kaufmann. "The Sorcerer's Apprentice: A serious game aiding rehabilitation in the context of Subacromial Impingement Syndrome". In: *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*. IEEE, 2013, pp. 327–330.

[2] Stephanie Jansen-Kosterink, Rianne M H A Huis in 't Veld, Christian Schönauer, Hannes Kaufmann, Hermie J Hermens, and Miriam Vollenbroek-Hutten. "A Serious Exergame for Patients Suffering from Chronic Musculoskeletal Back and Neck Pain: A Pilot Study". In: *Games for Health* 2.5 (2013), pp. 299–307.

[3] Annette Mossel, Christian Schönauer, Georg Gerstweiler, and Hannes Kaufmann. "ARTiFICe - Augmented Reality Framework for Distributed Collaboration". In: *The International Journal of Virtual Reality* 11.3 (2012), pp. 1–7.

[4] Christian Schönauer, Kenichiro Fukushi, Alex Olwal, Hannes Kaufmann, and Ramesh Raskar. "Multimodal Motion Guidance: Techniques for Adaptive and Dynamic Feedback". In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. New York, NY, USA: ACM, 2012, pp. 133–140.

[5] Christian Schönauer and Hannes Kaufmann. "Wide Area Motion Tracking Using Consumer Hardware". In: *The International Journal of Virtual Reality* 12.1 (2013), pp. 1–9.

[6] Christian Schönauer, Hannes Kaufmann, Stephanie Jansen-Kosterink, and Miriam Vollenbroek-Hutten. "Design eines Serious Games für die Rehabilitation von chronischen Rückenschmerzen". In: *Future and Reality of Gaming Vienna Games Conference, FROG 2012, Game Over. Was nun? Vom Nutzen und Nachteil des digitalen Spiels für das Leben*. Bundesministerium für Wirtschaft, Familie und Jugend, Abt. II/5, 2012, pp. 31–46.

[7] Christian Schönauer, Thomas Pintaric, and Hannes Kaufmann. "Full body interaction for serious games in motor rehabilitation". In: *Proceedings of the 2nd Augmented Human International Conference*. AH '11. New York, NY, USA: ACM, 2011, 4:1–4:8.

[8] Christian Schönauer, Thomas Pintaric, Hannes Kaufmann, Stephanie Jansen-Kosterink, and Miriam Vollenbroek-Hutten. "Chronic Pain Rehabilitation with a Serious Game using Multimodal Input". In: *Proceedings of Virtual Rehabilitation 2011*. 2011, pp. 1–8.

1.3.2 Technical Report

[1] Christian Schönauer, Thomas Pintaric, and Hannes Kaufmann. "Full Body Motion Capture - A Flexible Marker Based Solution". In: *Proceedings of Workshop on Accessibility Engineering with user models, simulation and VR*. 2012.

1.4 Dissertation Structure

This thesis is organized as follows. Chapter 2 gives an introduction to related research areas, such as key concepts of rehabilitation, input and output technologies used for serious games

in rehabilitation and game design considerations in this context. Chapter 3 briefly explains the theoretical foundations involved in optical marker tracking and motion capture. In Chapter 4 the development of our own flexible and accurate marker-based optical motion capture system including its methods, algorithms and workflow is described. Chapter 5 describes the design and development of a low-cost motion capture alternative based on RGB-D sensors and a flexible networked environment, that allows almost arbitrary scaling of the capture area. In Chapter 6 the design and development of a flexible VR/AR/multimodal feedback framework is detailed, including the integrated devices and examples of developed applications. A serious game based on the framework targeting rehabilitation of chronic pain is described in Chapter 7. Furthermore, details on a user study evaluating the game with ten patients are included. Finally, Chapter 8 concludes and summarizes this thesis.

CHAPTER 2

Related Work

The presented work touches many fields including serious games in rehabilitation and their design [112, 23, 80], full body interaction in conventional games [91, 34], MoCap systems [153], skeleton calibration [70, 115], tracking [136, 59], biofeedback devices [47, 157] and haptic feedback [83].

This chapter starts with a short introduction to the terminology in Section 2.1, attempting to define e.g. the very broad term "Serious Game". Section 2.2 follows up with key concepts and criteria that make serious games and virtual reality technologies good tools for rehabilitation. Input devices are an important factor of serious games, because they serve a dual purpose. On the one hand input devices allow interactive control of the game by the user, while on the other hand they measure patient performance from a medical point of view. The latter is important for feedback and evaluation of therapy progress. Therefore, Sections 2.3 through 2.6 give an overview of the state-of-the-art of input technologies. Section 2.3 provides a brief introduction to input technologies that have been used successfully in serious games for rehabilitation. In addition, Sections 2.4 and 2.5 describe full body MoCap technologies, that could be used as input to serious games, from a more general and technical perspective. Section 2.6 finally introduces biosignals and physiological measurements as input to serious games. Similar to the input devices, output technologies serve a dual purpose in the games. Firstly, they provide the user with game output and secondly they deliver feedback of performance to the patient (e.g. if a movement belonging to a rehabilitation exercise has been performed correctly). Consequently, Section 2.7 gives an overview of output modalities. Integrating various input and output devices into games and applications often significantly increases development effort. Therefore, OpenTracker, a framework providing flexible and transparent device integration is introduced in Section 2.8. Finally, Section 2.9 provides a short introduction to game design considerations specific to the development of serious games in rehabilitation.

2.1 Terminology

An important concept related to the work described in this thesis is "Virtual Rehabilitation", that according to the definition of Burdea "represents the provision of therapeutic interventions locally or at a distance, using Virtual Reality hardware and simulation" [21]. If the therapist is located remotely, it can also be categorized as "(Virtual) Telerehabilitation". Virtual rehabilitation has been successfully employed for motor as well as cognitive rehabilitation. It has shown great successes with various indications and treatments such as motor rehabilitation following stroke [57] or exposure therapy treating post traumatic stress disorder as shown by Rizzo et al. [116]. Furthermore, Lange et al. emphasize the "capacity of VR technology to create controllable, multisensory, interactive 3D stimulus environments, within which human performance can be motivated, captured and measured, offers clinical and research options that are not possible using traditional methods". Chapters 4 - 6 of this thesis describe the design, development and integration of VR input technologies as well as a framework facilitating the implementation of VR/AR applications and (serious) games.

Multiple definitions of the term "Serious Game" exist throughout literature. Originally, it has been coined by Abt in 1970 in his book [2], focused at education and not primarily aimed at video games, that were just emerging at that time. With the beginning of the Serious Games Initiative started by Sawyer and Rejeski in 2002 the term has been redefined. A newer and very broad definition of serious games is provided by Michael et al.: "games that do not have entertainment, enjoyment, or fun as their primary purpose" [90], which is widely acknowledged in related work. This definition incorporates games from various fields including training, health and education. The focus of this thesis is on serious games for health, specifically physical rehabilitation. An example for a serious game targeting chronic pain rehabilitation is described in Chapter 7. As in this thesis the fields of serious games and virtual rehabilitation are somewhat overlapping in many cases, since serious games often employ VR technology and virtual rehabilitation applications oftentimes have gamification aspects.

In the context of this work "Full Body Interaction" emphasizes input of all parts of the physical human body through MoCap and physiological measurements and feedback through visual and haptic modalities. Impairments and disabilities following stroke, traumatic brain injury or chronic pain include motor deficits like deterioration of the general condition or decreased quality of movement patterns and muscle coordination. Therefore, acquiring the full body motion data of the patient is the key for a multi-purpose system targeting more than a single specific body part. The related term "Whole Body Interaction" coined by England [37] focuses also on emotion and social context and therefore it is bit too wide for our purpose.

Input and output devices are a significant part of serious game systems in motor rehabilitation. During conventional occupational or physical therapy, a therapist is observing and guiding the patient in his relearning process and corrects errors in his movement patterns. In serious games or virtual rehabilitation, especially if considering telerehabilitation, the system or game has to give this feedback to the patient. The requirement to provide a user with feedback in a technical systems places certain requirements on the measurement and feedback technologies. For knowledge of performance feedback has to be provided in real-time, thus measurements and their processing have to be fast. In order to capture the progress of a patient throughout therapy,

thus providing knowledge of results, measurements have to be precise and repeatable. Both criteria a met by the technologies described in Chapters 4 and 5 and Section 6.6 and the framework described in Chapter 6.

2.2 Key Concepts in Rehabilitation and How They Can Be Targeted in Serious Games

Holden [57] provides a scientific rationale for the use of VR technology in rehabilitation and discusses three key concepts relevant to motor learning in that context (motivation, repetition and feedback). However, Holden's original work [57] neglects somewhat the concept of motivation in her theoretical discussion and review. The next three subsection provide a short introduction to these key concept from a perspective of serious games in rehabilitation. Furthermore, Rego et al. [112] have formulated criteria (e.g. adaptability, performance feedback, progress monitoring) for making serious games for rehabilitation more functional tools. We have considered these concepts and criteria throughout our design as described in Chapters 4 and 7.

2.2.1 Motivation

In many cases rehabilitation is a procedure that patients have to actively pursue for a long time, sometimes for the rest of their lifes (e.g. for chronic pain rehabilitation). Therefore, patients often lose motivation in the process. Some might not even finish their treatment program due to the repetitive nature of the tasks [112, 142]. Increasing the patient's motivation is one of the obvious reasons for using games in serious applications. Patients are encouraged to exercise, because the medium of computer games has been described as having the potential for being very engaging, even addictive for users [23]. Therefore, a challenging game environment can be considered to be motivating [78], but also distracting [66], e.g. from pain associated with certain conditions. In physical rehabilitation, patients are required to undergo training protocols that are experienced as being boring due to the high level of repetitions. Games can help to transform an often repetitive, monotone training task into a more engaging experience [3]. In addition, serious games are interesting for rehabilitation, because the proposed exercises should be done frequently and on an intense level, which can be supported very well by the motivating nature of game media [23]. Thus, games challenge participants to play repeatedly to beat their personal high score and thereby increase their treatment intensity. Furthermore, Roy et al. emphasize the importance of feedback especially for effective unsupervised home training [118]. This feedback can be provided in various forms in a game, thus increasing motivation.

Motivation is especially important in home-based or telerehabilitation scenarios, where serious games are an attractive addition to conventional face to face physical therapy [13]. Without appointed times and scheduled therapy sessions, patients often find it difficult to stick to the prescribed exercises. Here a technical system can help the patients to hold on to the therapy plan in absence of a therapist. At the same time the patient maintain individual responsibility for their therapy. Therefore, games have the potential to increase the treatment compliance in physical rehabilitation.

2.2.2 Repetition

Repeated execution of movements is an important factor for motor learning and the associated cortical changes. However, practice has to be linked to (incremental) success in a task in order to be effective [124]. Therefore, the linkage between the repeated exercise and provided feedback is crucial. Also neurophysiological evidence shows that "skilled movements" - as opposed to blind repetition of simple movements - are crucial to the success of motor learning [112]. Therefore, the repetitive nature of games is thought to be a key mechanism that promotes motor skills learning. Furthermore, games have the potential to positively influence physical performances [49] in populations of medical patients, especially those with physical impairments [66].

2.2.3 Feedback

The third important factor for patients in rehabilitation is feedback on the way they perform and the results of their exercise. From literature we know that feedback learning curves are higher when subjects have knowledge of performance (KP) compared to subjects who have knowledge of results (KR). Combining KP and KR is even more powerful [124]. Thus patients should get feedback on how they perform during and directly after the end of the exercise. Feedback based on knowledge of performance will also motivate the patient to reach the goal. For this feedback loop the system has to be able to provide accurate and timely measurements, thus requires to integrate appropriate input, tracking and output technologies. In this context Lange et al. describe the task of tracking the user as "primary challenge for virtual rehabilitation" [80] and that "Physical rehabilitation requires accurate and appropriate tracking and feedback of performance" [79].

The concept of "providing biological information to patients in real-time" is also known as biofeedback and has been successfully applied for more than fifty years [47]. Giggins et al. categorize biofeedback measurements in physical rehabilitation into biomechanical and physiological. Biomechanical biofeedback relies on measurements of movement, postural control or force, while physiological biofeedback can be provided on neuromuscular, cardiovascular or respiratory data. The focus of this thesis is on feedback from biomechanical measurements especially movement and to a lesser degree on physiological measurements. An introduction to devices capable of delivering data for movement biofeedback is therefore given in Sections 2.3 - 2.5, while physiological biofeedback is introduced in Section 2.6.

2.3 Input Devices for Serious Games in Rehabilitation

Various input devices have been used to provide input to serious games in rehabilitation: from technology available in commercial gaming systems [34, 91], to specialized input devices like a pressure mat [12], to webcams [23] and more sophisticated technologies like data gloves and magnetic tracking [85]. The trend for the input in conventional, as well as, serious games goes towards full body interaction. Microsoft's Kinect [91] tracks the player's whole body movement and integrates this body state information into the game. Using Primesense's camera/depth sensor and associated middleware like FAAST [143] full body interaction has become newly available for application of serious games in rehabilitation. Recent publications have successfully

demonstrated the feasibility of using Kinect in serious games for rehabilitation and good compliance with patients and therapists [79]. Due to its technological limitation, however, Kinect and similar sensors lack the accuracy required for a more detailed data analysis in some medical applications. In Chapter 5 we describe the integration of these low-cost tracking devices in our system and present a brief technical evaluation comparing Kinect and the MoCap system introduced in Chapter 4.

2.4 Full Body Motion Capture Systems and Technologies

Currently, four groups of systems are frequently used for full body MoCap, based on optical, magnetic, mechanical and inertial technologies. Acoustic tracking plays a subordinate role for MoCap systems, but sometimes is used in hybrid setups e.g. in combination with inertial tracking [156]. Furthermore, marker-less MoCap systems as described in the next section have strongly increased in quality during the last couple of years [42].

Mechanical MoCap systems offer a simple and fast way to obtain the pose and movements of a human user. For the tracking process an artificial articulated skeleton is built around the body. The angles of the artificial joints are then being calculated using measurements of sensors (e.g. potentiometers or fiber-optic sensors). From the relative orientations between bones forward kinematics can be used to calculate the pose of the body using a hierarchical model of the body. Mechanical systems are robust to magnetic and electrical interferences and not susceptible to occlusion. However, the physical exoskeleton limits mobility of the user and poses restriction on movements [22, 18].

Magnetic MoCap systems use a magnetic field created by a stationary transmitter to measure the 6DOF pose of multiple receivers placed on the user's body. Usually three orthogonal coils are excited with either AC or DC currents generating either oscillating or pulsed magnetic fields. On the receiver side either magnetometers or hall effect sensors (AC) or orthogonal coils (DC) and associated electronic units measure field strengths. A discussion of specifics of both approaches can be found in [22]. The main advantage of magnetic tracking systems is the out-of-the-box 6DOF pose and robustness to occlusion of non-metallic objects. On the other hand, the capture volume is limited by the magnetic field strength and subject to magnetic and electrical interferences. Magnetic tracking has become recently available as a consumer product as well with the Razer Hydra [60]. However, it is intended as desktop device providing 6DOF tracking of two controllers within about arm-length distance of a base station. Nevertheless, it provides interesting options for interaction and we have integrated it in our framework described in Chapter 6.

Xsense [165] and other companies, offer solutions for MoCap using inertial tracking, that however, are also relatively expensive, especially with enough sensors for full body configurations. Furthermore, these systems suffer from error accumulation due to sensor drift [161]. Nevertheless, the big advantage of inertial sensors is the possibility of larger capture volumes. With the availability of low-cost accelerometers within the recent years, MoCap solutions based on inertial technology are currently being developed, that are targeting the gamers as user group, such as the Perception Neuron MoCap system [103]. However, until now these consumer products have shown to be significantly inferior to professional setups.

Marker based optical full body MoCap (e.g. [153, 94]) has become a de facto standard capture technology in the movie and entertainment industry in the previous 10 years. Thereby, one or more actors wearing MoCap suits with retro reflective markers attached to them, are tracked by a number of cameras. All motions are computed in real time (with millimeter accuracy) and are available for further processing e.g. recording, analyzing, motion transfer to a virtual character and more. These setups can be scaled to cover huge areas with multiple users by using large numbers of cameras. In a simplified form (i.e. with a small number of markers per user) these systems have also been widely deployed for medical applications such as gait analysis. However, marker based systems have not been used for full body input to serious games in rehabilitation due to numerous reasons like cost, inflexibility and complexity of operation, which are issues we are targeting with our system detailed in Chapter 4.

Finally, MoCap systems based on markerless optical tracking have become available in recent years. Although they are promising in terms of availability, cost and usability, they in many cases lack the accuracy provided by the traditional MoCap technologies and are therefore described separately in the next section.

2.5 Markerless Motion Capture

A large effort in computer vision research has been put on marker-less human MoCap from camera RGB-images [35]. Recent successes [42] have resulted in the emergence of MoCap systems capturing not only the movement of a skeleton but also 3D surface deformation in real-time [148]. However, this is still a difficult problem in terms of reliability and accuracy, especially, if only few cameras are used. Therefore, in previous years the use of time-of-flight and other depth cameras for MoCap has become an active research area [72, 106]. Only recently, this trend has extended to the use of depth cameras in games and entertainment. Multiple products have been released using structured light projection together with an infrared camera to acquire depth information [7, 91, 138]. Together with these sensors, software bundles are available, which provide human posture data calculated from the depth images in real-time e.g. using the algorithm described in [135]. While structured light has been used for shape reconstruction before [56], these systems offer entirely new possibilities for entertainment, technical enthusiasts and scientists. Multiple research applications have been developed. These are using the depth information for e.g. facial animation [160] or posture information for e.g. gesture recognition [143]. With an easy-to-use hardware setup and reasonable computational effort, cheap MoCap solutions have made it into the living rooms of users and provide input to console and PC applications. For many games, these systems are sufficient in terms of tracking quality and capture volume. However, with a view towards ubiquitous computing [107, 162] and setups other than the classical home entertainment center (with a TV placed in front of the couch in the living room), improvement is needed. Work has already been conducted on merging multiple depth cameras to increase the volume of interaction and compensate for occlusions [162]. However, these approaches do not use full body tracking and are still limiting interaction to a room-sized environment. We have therefore designed and developed a system based on multiple RGB-D sensors, that can be flexibly combined to provide MoCap data of larger environments, as described in Chapter 5.

2.6 Physiological Biosignals in Rehabilitation

In recent years biosignals have become increasingly important as a non classical user interface. Especially in medical applications, where users' physiological measurements are of vital importance, various sensors have been integrated and used for biofeedback to patients and medical personnel [47, 157]. Giggins et al. state in their review [47] that neuromusculuar measurement using electromyography (EMG), i.e. muscle activity, is the most widely reported form of biofeedback and appears promising. In serious games, however, physiological measurements like EMG have been mainly used to measure engagement and excitement [132]. Therefore, we have integrated biosignal acquisition devices providing EMG measurements into our framework described in Chapter 6 and used this data as input to the serious game introduced in Chapter 7.

According to Giggins et al. biofeedback can be relayed to the user as either direct or transformed feedback. Direct feedback would e.g. display the heart rate as numerical value, while transformed feedback presents the user with visual, audio or tactile content based on the measurement. For the use in serious games such as the one described in Chapter 7 usually transformed feedback will be used, because game elements allow for a large variety of ways to interpret measurements.

Physiological measurements recorded during the use of serious games in rehabilitation could be helpful even beyond providing real-time feedback to the user. Collecting data as a side product of a serious game might help to increase efficiency and applicability of existing technology, as well as provide deeper insights in different medial indications.

2.7 Output Modalities

According to Rego et al. interaction technology and performance feedback are two of the main criteria for serious games in rehabilitation [112]. Besides the input devices described in the previous sections, various output technologies are the second crucial part of an interactive system and the feedback loop. We perceive our environment with multiple senses simultaneously, as real-world feedback is multimodal. To simulate this in a VR/AR system, different output modalities and actuators have to be employed. We provide a brief overview of modalities and attributes with a view towards VR/AR setups and motor learning and motion guidance tasks relevant in rehabilitation setting.

Visual feedback is considered the dominant form of feedback [62]. Besides or through 3D content of a serious game it can provide users with objective observation of their motion (errors), augmented awareness for correction (e.g., emphasized body parts indication), superimposed target trajectories, or scores [27, 95]. Chapter 7 provides an example how visual and audio feedback can be integrated in a serious game for rehabilitation.

Usability and user experience depend on the display hardware and properties, including latency, update rate, size, resolution, mobility and stereoscopy [95, 18]. An extensive review of display technologies is beyond the scope of this thesis and can be found in [18] and [22]. We can categorize VR/AR setups according to the degree of immersion produced by the display hardware and to a lesser extent by other (input) technologies. Non immersive systems, which are also often referred to as desktop systems, usually use (non stereoscopic) monitors or small scale

projections to display the virtual environment. The user remains stationary in front of the display interacting usually through standard desktop input devices or 3D interaction devices, e.g. a 3D mouse or specialized 3D desktop input technology such as the Razer Hydra [60] (see Section 6.6.1 on details). The big advantage of these systems is the availability and comparatively low price, which makes them interesting for use in applications where cost is a limiting factor (e.g. telerehabilitation or education as described in Section 6.9.2).

Semi immersive systems mostly rely on large scale (stereoscopic) projections with high resolutions, thus increasing the feeling of immersion and proving a better sense of depth. For interaction often the tracking technologies described in Section 2.4 are used. Examples for semi immersive systems are presented in Sections 6.9.3.2 and 7.5.1.2.

Fully immersive systems usually employ head mounted displays (HMD), e.g. [99], other head coupled devices such as the Binocular Omni-Orientation Monitor (BOOM) [15], or the CAVE Automatic Virtual Environment (room sized environment with multiple walls covered with projections) [30]. Again the tracking technologies detailed in Section 2.4 are usually applied for interaction. An example of a fully immersive system is described in Section 6.9.3.3. Technical development in the recent years are about to make available fully immersive setups to a wider public with consumer HMDs such as the Oculus Rift [99].

Chapters 6 and 7 present a number of setups with a varying degree of immersion and different display technologies.

Spatial Audio can be used to communicate 3D positions, direction or give more abstract feedback on other performance parameters or events. Dozza et al., for example, encode information using sinusoids with variable frequency and loudness [36]. An example of how audio feedback is being used in a serious game can be found in Section 7.3.

Haptic feedback is composed of skin sensation and proprioception [62]. **Skin sensation** is perceived by sensory receptors that enable us to localize and recognize individual tactile cues. Tactile displays can often indicate body part positions more intuitively than visual or audio feedback [139, 82] or deliver contact cues in virtual environments to increase situational awareness. Robustness and low weight has made vibration motors popular for tactile actuation. Baek et al. [9] present a training system for fencing and dancing, and Rosenthal et al. explore foot step navigation in dance [117]. Spelmezan et al. [139] study spontaneous reactions to tactile patterns, while TIKL [82] gives a user visual information and vibrotactile feedback on the deviation of arm posture. McDaniel et al. [89] propose a framework for mapping vibrotactile patterns to motion. Pneumatic pressure could potentially provide well localizable high-intensity stimuli, but is less popular due to size and tubing complexity of the required compressor. Finally, skin stretching with customized actuators may work well for indicating directions as pin-based actuators enable fine-grained feedback to the human skin [62]. Constructions are, however, often bulky. Sections 6.5.2.7 and 6.6.6 describe the design and integration of our tactile displays based on vibrotactile and pneumatic actuators.

2.8 Tracking Middleware/OpenTracker

2.8.1 Design

OpenTracker [114] is an open source (tracking) middleware, which offers transparent and flexible device integration, transforms input events and passes them through a multi-modal data flow network to an application. It is framework providing functionalities required for tracking in VR/AR applications and easing the development and maintenance of hardware setups. Furthermore, it can transform and extract data in a flexible and reusable manner.

The data-flow network can be interpreted as graph, where each of the nodes specifies a unit of operation. The main types of nodes are source, sink and filter. Source nodes, which are leaves in the graph, usually receive their data from devices or other external sources. Filter nodes are internal nodes and manipulate or extract data (e.g. geometric transformations, smoothing filters, merge filters, etc.), and sink nodes, which propagate data values to an application or write it to another external output (e.g. network, file). The basic concept of the data-flow network is that of passing events between the nodes. Therefore, the nodes can be connected by directed edges over which timestamped events are sent. Each node can have multiple incoming edges, but only one outgoing edge is permitted. This allows for e.g. merging of multiple tracking sources or parameterization of computations, while the structure remains clear. The events passed through the data-flow network can contain various data flexibly configurable by the user.

2.8.2 Implementation

OpenTracker provides an object-oriented design and configuration through XML files. The user utilizes standard XML tools for development, configuration and documentation. OpenTracker allows multi-threaded execution and filters apply transformations to tracking- and other input data [114]. The class central to OpenTracker is *Context*, which runs the main loop, configures and manages all modules and the data flow. At startup *Context* parses the XML based configuration files used to describe tracking configurations, that usually consist of multiple input devices, transformations and one or more output targets. The nodes of the data-flow network are dynamically instantiated. The *Context* can be run in a separate process or from a thread in the application as described in Section 6.5.1.2.

We have made use of several of the standard nodes provided in OT in our ARTiFICe framework described in Chapter 6, which are briefly introduced in the following. To fetch tracking data from remote input devices, Virtual-Reality Private Network (VRPN) [147] can be used. It is a device-independent and network-transparent framework for devices used in VR/AR systems. OpenTracker implements a *VRPNSource* and a *VRPNSink* node. *VRPNSource* can connect (over network) to a device providing data through a VRPN server and insert the tracking data as event into the OT data-flow network. *VRPNSink* on the other hand runs a VRPN server instance and allows other applications implementing the VRPN client interface to connect and receive the data. VRPN has been used to connect tracking and MoCap devices and tools as described in Sections 6.6.2, 6.6.3 and 6.8. In addition, OT integrates a biosignal acquisition system, the Gtec g.MOBIlab [50]. We have extended and used the *MobilabSource* node as described in Section

6.6.4. Furthermore, OT supports the tracking library ARToolKit, that targets Augmented Reality setups, throught the node *ARToolKitSource*, as detailed in Section 6.6.1.

2.8.3 Extending Opentracker

OpenTracker provides a number of source, filter and sink nodes, but can be easily extended by specialized classes deriving from the provided base classes.

Source Nodes Source nodes usually encapsulate a device driver and insert events into the data-flow network based on the data received from the device. We have added the following new source nodes to OT.

IOTMCSource wraps the MoCap system introduced in Chapter 4 into OT as detailed in Section 6.6.3.

TMSISource integrates a new biosignal acquisition system, the TMSi Mobi as described in Section 6.6.4.

SpaceDeviceSource adds support for the 3D mouse 3D Connexion SpaceNavigator as introduced in Section 6.6.1.

HydraSource implements a source node for the Razer Hydra 6DOF desktop interaction device as detailed in Section 6.6.1.

Filter Nodes Filter nodes process and transform received events and propagate the calculated data further into the data-flow network.

IOTMCSkelMod is a filter node, which calculates important movement parameters (e.g. walking velocity) and specific positions on the skeleton as described in Sections 4.3.3 and 6.6.3

HighPassFilterNode and *RMSFilter* implement high pass and root mean squares filters used for processing EMG data as detailed in Section 6.6.4.

HeartrateFilter implements a filter node, which calculates the heart rate based on the ECG signal received from a biosignal acquisition device and is introduced in Section 6.6.4.

Sink Nodes Sink nodes distribute events outside the OT data-flow network. Output data can be written to an application over shared memory, over network or to file.

UnitySink has been implemented to stream data into Unity3D over either shared memory or a network interface as described in Sections 6.5.1.2 and 6.5.2.3. *UnitySink* is the main interface between OT and the application layer of our ARTiFICe framework introduced in Chapter 6.

2.9 Game Design Considerations and Serious Games in Rehabilitation

Serious games in motor-rehabilitation are still in their infancy, but have become an active research area within the last couple of years [69]. They have been used to assist therapy following stroke, traumatic brain- and spinal cord injury and in other areas. Also the focus of therapy is very diverse, such as upper limb rehabilitation [23], balance rehabilitation [12] and many others [112]. Specific requirements and design principles apply to serious games in motor rehabilitation [112, 23]. Although promising, commercial exergames are designed primarily for entertainment, not specifically to train impaired patients in a professional rehabilitation setting [146]. These games are more used as a pleasurable alternative instead of being a proper tool for physical rehabilitation. Practice shows a number of limitations when applying existing commercial exergames into professional rehabilitation settings. Firstly, commercial exergames are developed for a general public and the norms are derived from the performance of healthy gamers. In most cases, it is not possible to change these norms and to personalize the game for an impaired patient. The game becomes too hard to play which can lead to frustration. Secondly, the level of energy expenditures reported for commercial exergames cannot be compared to levels of energy expenditures reached during physical therapy [49] and the requested motor skills are too general and not pathology specific. Thirdly, the game output of commercial exergames focus on the game performance, e.g. time or high score, and not the patient's performance on motor skills and the physiotherapist cannot monitor the progression of the patient on the performed motor skills. Therefore, serious games for rehabilitation should provide configurability to the patients' abilities, include exercises specific to the targeted indication and provide means to record and monitor data on patient performance, which is also emphasized by [80]. While almost all the serious game systems offer options for adjustment of general game parameters like difficulty, speed and number of repetition [12, 23, 34, 85], options for adaptability of specific input measurements are often limited. Thus, the games cannot be adapted to certain handicaps a patient might have or adapt to the progress during therapy only in a very limited fashion. In our game described in Chapter 7 configurability has been emphasized and all parameters measured by MoCap and biosignals can be adapted (as baseline and therapy goal) in addition to in game parameters. In existing serious games for rehabilitation, measurements recorded for further evaluation seem limited or are not documented.

The core idea of applying games in the context of rehabilitation is of course to provide motivation by making it "fun" for the patients. A discussion of the term "fun" in the context of games and how it can be achieved through game design choices can be found in [74]. In games fun is often created by the mastery of a challenge [29, 74]. Koster emphasizes that challenges have to precisely meet the player's skills to achieve "flow", fun and avoid boredom and frustration. This in turn again shows the importance of configurability, as pointed out earlier in from a medical perspective. Burke et al. identified meaningful play in addition to challenge as important principle for the design of games in rehabilitation [23]. Meaningful play results from the relationship between a players action and the systems response to this action. This relationship should be meaningful discernable (perceivable by the user) and integrated (it affects the game experience at later moments). A detailed introduction to the concept of meaningful

play can be found in [119].

Most serious game systems in rehabilitation have not evolved above prototype stadium and tests with a few patients or healthy subjects. Therefore, often a closed workflow is missing, which should reach from (1) calibration of input to (2) configuration of the game to (3) gameplay to (4) patient feedback and finally (5) medical evaluations. One system, which has been established in many clinics and studies, is RehaCom [113]. It provides such a workflow. However, it is limited to cognitive rehabilitation and standard input devices (mouse/joystick). In our system we have established such a workflow, which should enhance efficiency throughout the evaluation process as described in Section 4.1.

CHAPTER 3

Theoretical Foundation

This chapter provides details on the theoretical and algorithmic background of this thesis, mainly those related to marker-based optical tracking and motion capture.

3.1 Optical Pose Tracking

Burdea et al. define an optical tracker as "a noncontact position measurement device that uses optical sensing to determine the real-time position/orientation of an object" [22]. Thus, the tracker produces the pose of a tracked object with 6 degrees of freedom.

The MoCap system presented in Chapter 4 builds upon such an optical tracker - the iotracker [105] - and therefore hardware, concepts and algorithms of this system are briefly introduced here. Most optical trackers, as-well-as the iotracker, are outside-looking-in, i.e. with the sensors fixed and some light beacon on the user [22]. Therefore, the following sections will focus on the principles of such systems, although other (inside-looking-out) exist. Section 3.1.1 introduces the iotracker setup, which is exemplary for that of an outside-looking-in tracker. Figure 3.1 shows the basic pipeline of an optical tracker. Based on the video input of the trackers cameras the following processing steps are executed:

- Feature Segmentation (Section 3.1.2)
- Projective Reconstruction (Section 3.1.3)
- Model Fitting (Section 3.1.4)
- Pose Estimation (Section 3.1.5)

The steps are described in the following sections with an emphasis on model fitting in Section 3.1.4, because it is the most relevant for the MoCap algorithm described in Chapter 4. More details about all steps can be found in [105].

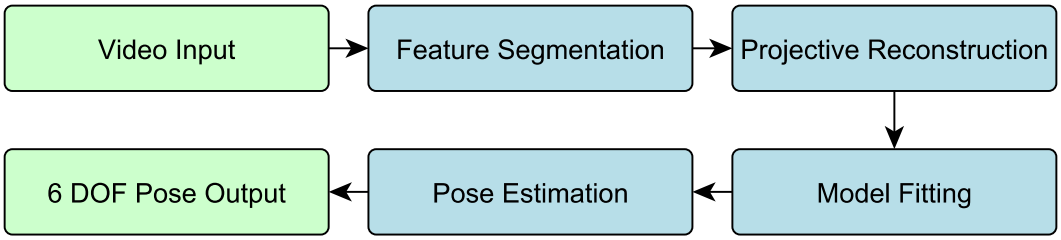


Figure 3.1: Basic pipline of an optical outside-looking-in tracker.

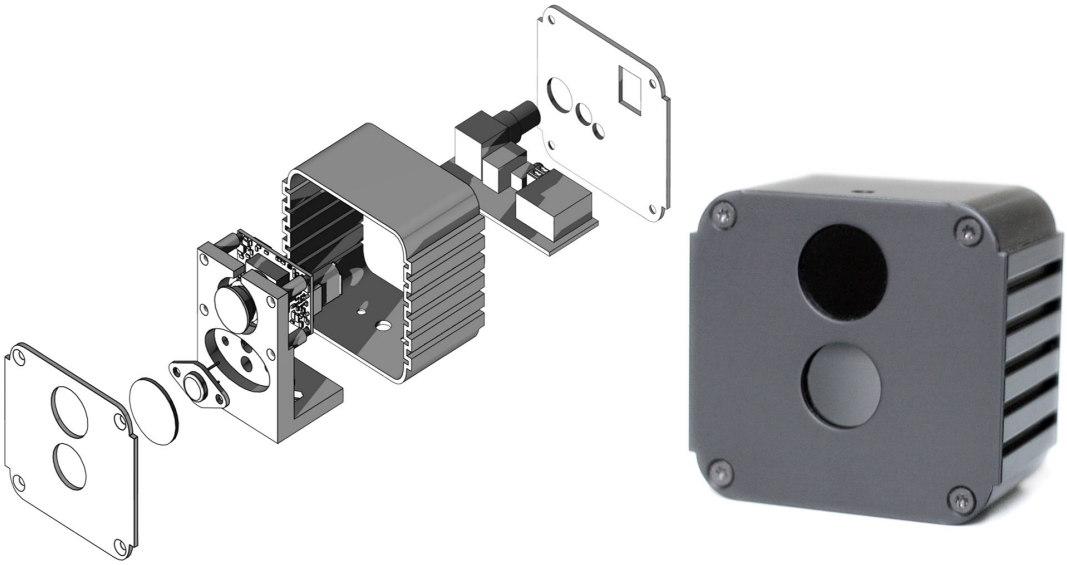


Figure 3.2: Explosion chart and picture of the camera used in our system.

3.1.1 The iotracker - Infrared Optical Marker Tracking Setup

The tracker used for our motion capture system is an iotracker [105], which is a passive marker based infrared optical motion tracking system. It was primarily built for collaborative virtual reality and augmented reality applications, but is also well suited for motion capture. Commodity hardware is used to minimize the cost and calculations are performed on PC workstations, while no additional camera electronics is required. Iotracker cameras are shutter synchronized and stream digital video to the tracking workstation at an update rate of 60 Hz. They have a very compact form factor of about 7x7x4 centimeters. To make (only) the passive markers visible in the camera images the cameras are equipped with infrared strobe lights and optical band pass filters. Usually, an iotracker setup consists of four to eight cameras illustrated in Figure 3.2 with all components. The cameras are calibrated relative to each other and produce the video input for the pipeline illustrated in Figure 3.1.

3.1.2 Feature Segmentation

In the first step of the pipeline features are identified in the synchronized video images. After masking out background light and reflections in a calibration step, at runtime only the spherical markers should be visible as bright round blobs of pixels. These blob features are segmented and their centroids calculated. The iotracker software segments the blobs using binary thresholding, calculates the centroids based on luminance-weights and recovers the centers of overlapping blobs with circular Hough transform [105]. The result is then passed on to the next step of the pipeline to calculate the 3D positions of the markers.

3.1.3 Projective Reconstruction

Based on two or more 2D positions of a marker in the camera images its 3D position can be triangulated. However, with multiple passive markers the correspondences between the blobs in different images are not known. Therefore, multiple view feature correlation has to be performed. The combinatorial problem can be reduced using observations from epipolar geometry as described in [105]. Correspondences between marker features in different views are selected based on the reprojection error. From the selected candidate features the algorithm triangulates the 3D marker position.

3.1.4 Model Fitting

In order to track a 6DOF pose of an object, the 3D positions of at least three markers rigidly connected to the object have to be known. These rigid configurations of markers are often called targets and in the case of the iotracker consist of four markers as shown in Figure 3.3. The iotracker attempts to uniquely identify markers of each target within a larger, unstructured set of observed marker 3D positions. Therefore, it compares the Euclidean distances between all pairs of observed 3D positions with the known distances between markers of each target. Consequently, the iotracker targets are assembled in a way to maximize differences of distances between the markers on the same target as well as between sets of multiple targets.

To account for the presence of noise and different types of process errors, two distances d_i and d_j are considered equal if $|d_i - d_j| < \epsilon$.

The problem of finding targets in the point cloud can be seen as a case of the maximum clique problem. Maximum clique is a well-studied problem in graph theory for which fast algorithms exist [102]. Nevertheless, it is np-hard, which is why a polynomial-time complexity reduction is applied, that is based on a form of geometric hashing.

3.1.4.1 Complexity Reduction

At startup, the Euclidean marker distance matrix is computed for each target and its entries stored in a one-dimensional look-up table (Figure 3.4 first line). These tables are used at runtime to rule out unlikely clique-to-marker matches before the maximum clique problem is solved.

According to the evaluation presented in [105], absolute marker tracking errors have an approximately Gaussian distribution. Uncertainties calculated from the measured distribution's standard deviation are integrated with the tables as shown in Figure 3.4. The entries stored in

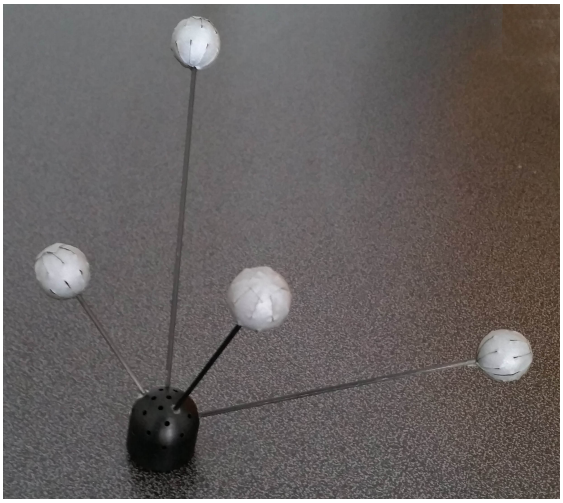


Figure 3.3: Picture of an iotracker target composed of four markers.

this look-up table are referred to 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.

With each new frame, the Euclidean distances for all combinations of 3D marker positions are calculated creating a matrix (Figure 3.4 second line). For every distance value in the matrix, the algorithm looks up the correlation scores in the tables described above and stores it in a separate correlation score matrix. Then, row-wise summation of the matrix's elements produces correlation scores of each marker in vector form. The values contained in the vector can then be thresholded to determine if a marker is likely to be part of a certain clique or not. The calculations necessary for this algorithm can be parallelized very well and significantly reduce the number of possible marker correspondences.

3.1.4.2 Clique Search

During the previous complexity reduction step a large number of markers are ruled out to be part of the solution set. However, usually more markers than the targeted clique size (usually four) remain candidates. These candidates for a clique are interpreted as nodes of a graph connected by undirected edges. On this graph the maximum clique search is performed to retrieve vertex cliques. If just one clique is found, which can be considered the normal and ideal case, the algorithm is finished. If multiple possible cliques are found, the square sum of marker-to-model alignment errors is used to determine the best solution. Also predicted poses from previous frames can be used to select a clique. If no solution for the desired vertex number can be found, but there are more than three vertices left in the original graph, the search is restarted for smaller cliques with at least three markers (a minimum of three markers is needed to determine 6DOF in the next step).

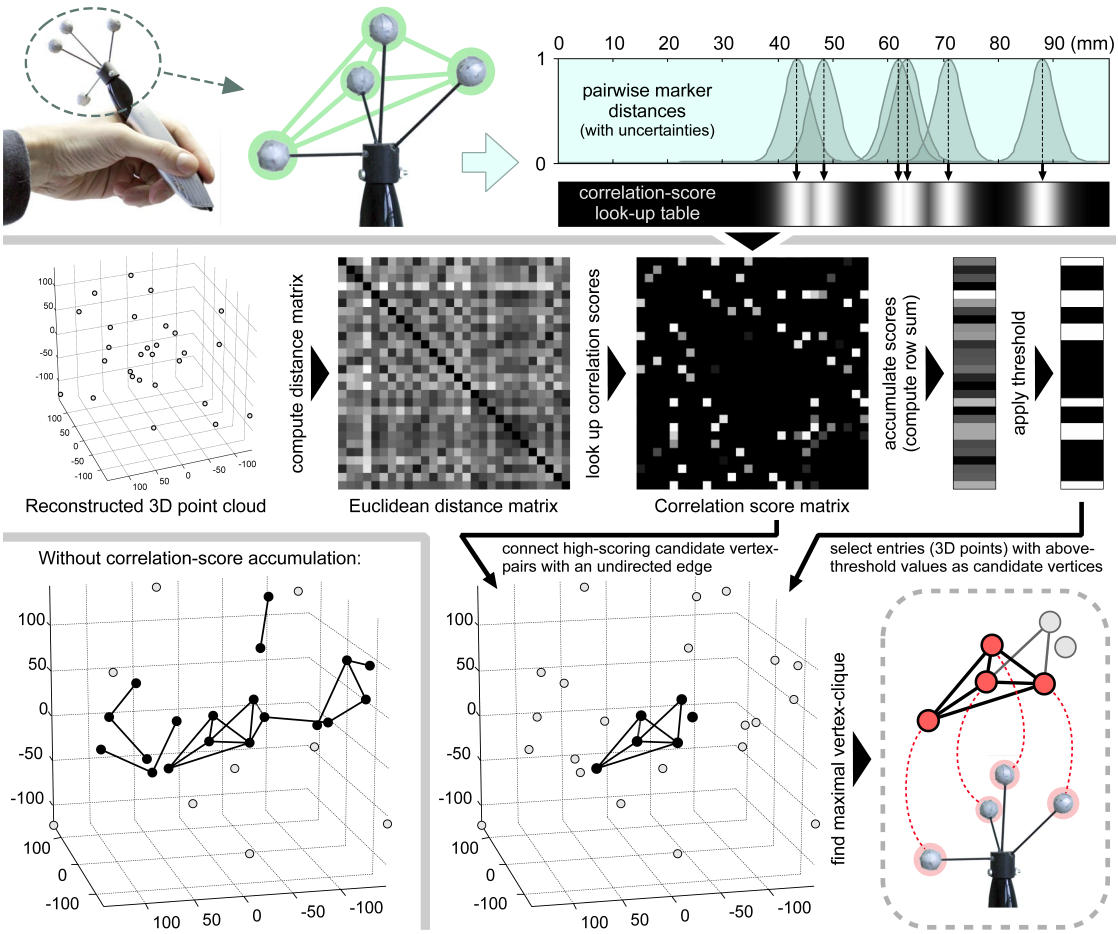


Figure 3.4: Quadratic-time complexity reduction (top). Formulation of the model-fitting problem as maximum-clique search (bottom). (Source: [105])

3.1.5 Pose Estimation

Once the maximum clique problem is solved, the clique can be aligned with the data by minimizing the least squares distance between observed markers and the clique's idealized markers as stored in the model. This distance value is also used as error function (model alignment error) in the other steps. The 3D transformation necessary to align the model correctly can be calculated using a closed-form solution (e.g. as described by Horn [58]). The pose is the output of this step and can be subjected to filtering if necessary.

3.1.6 Predictive Filtering

Predictive filtering can help to reduce jitter and latency of the tracking and even compensate for very short occlusions of markers. The pose of tracked objects is therefore subjected to Double Exponential Smoothing Prediction (DESP) as described by LaViola [81]. Exponential

smoothing filters a time series $p_0, p_1, p_2, ...p_t$ by calculating the filtered value by $Sp_t = \alpha p_t + (1 - \alpha)Sp_{t-1}$. The weights put on different measurements decays exponentially over time with $\alpha \in [0, 1]$ parameterizing the speed of decay. Double exponential smoothing applies exponential smoothing twice, which is expected to be better suited to follow trends in human motion [81]. Thereby, DESP models a given time series using a linear regression equation and then applies exponential smoothing not only on the values of the time series but also the parameters (y-intercept and slope) of this equation. The trend captured by the evolving parameters can then be used to make predictions of position and rotation into the future.

3.2 Algorithms for Marker-Based Infrared Optical Motion Capture

In general, a MoCap system has to solve two main problems in order to determine the posture of a user. First, a model, matching the current user's physical properties, has to be created. Then, this model must be fitted to the tracking data in each frame in order to determine the user's posture. In the case of marker-based infrared optical MoCap, the data is an unstructured point-cloud of marker positions.

Our system solves both problems with a minimum of user interaction, while providing posture information in real-time.

Not much information is available on the intrinsic functioning of commercially available systems and how they create or adapt a skeleton model for a specific user and fit it to the marker positions during the tracking process. This thesis and the related publications attempt to fill this gap. Chapter 4 describes the applied algorithms and methods that were used to achieve these goals in order to develop a highly accurate, commercial grade, robust optical motion capture system.

3.2.1 Skeleton Calibration for Full Body Motion Capture

Current commercially available full body MoCap systems (e.g. Vicon [153]) and research prototypes often rely on a predefined skeleton model with a certain number of markers, which have fixed positions on the tracked subjects [59]. This reduces the flexibility of adaptation to different users and use for more specific applications only needing input from a part of the body. Furthermore, precise placement of markers requires expertise and hinders easy application.

Therefore, in our system the skeleton of a user is calibrated from arbitrary (albeit within certain constraints) marker placement.

Several methods have been proposed for the automatic calculation of a skeleton model from marker-based motion capture data. Often, they rely on the calculation of the center of rotation between segments defined by their 6 DOF pose. To track position and rotation of a segment three or more optical markers [136, 26, 115] or magnetic tracking [98] have been used in this context. In these approaches using two adjacent segments with 6 DOF the joint positions are calculated using a least squares fit. However, these methods require many markers or (inaccurate) magnetic tracking (with one sensor attached to each segment). The method described in [70] suggests using a computationally more expensive algorithm, which requires less markers. Since the basic

method by Kirk et al. is essential to our calibration algorithm, it is described in the following subsections. An unshortened version of this description can be found in [125]. Details on how we have improved the approach to be more robust and reliable can be found in Section 4.2.

The basic steps of the algorithm are the following

1. Marker correspondence - assigning the markers ids (Section 3.2.1.1)
2. Marker clustering - group markers on a segment (Section 3.2.1.2)
3. Joint position calculation (Section 3.2.1.3)
4. Computation of skeleton parameters (Section 3.2.1.4)

3.2.1.1 Marker Correspondence

For most skeleton calibration algorithms including the one of Kirk et al. [70] correspondence of markers between frames has to be identified. Many current approaches rely on marker data from systems with active markers [70], where finding correspondence is trivial, because the system delivers the marker-ids, e.g. through light modulation [104]. Optical trackers using passive markers don't relay the marker identities to the system. They just produce a cloud of points for each frame and correspondences have to be found from frame to frame. As long as the tracking frame rate is high enough and the markers don't move too fast, positions of markers will not differ by much in adjacent frames. Therefore, in simple scenarios correspondences can be easily established by labeling close marker positions in successive frames as belonging to the same physical marker.

In practice, however, markers get lost and reappear during the tracking process, due to temporal occlusion by clothing or limbs. To retrieve the identity of a reappeared marker, however, Kirk et al. suggest a more sophisticated algorithm.

A marker that reappears, after it has been occluded, is treated as if it were a new marker and assigned a new identity. Thus, one physical marker on the actor's MoCap suit can have multiple identities over the captured time span. As long as that only happens to very few markers, it could very well be ignored and every identity treated as if it were a marker on the tracked subject. For more markers disappearing/reappearing, however, this results in a degradation of the following stages's accuracy. Therefore an algorithm is needed to find all those identities, that belong to one marker, and merge them into a single data set.

For this purpose [70] suggests a method that exploits the fact, that the tracked subject often moves through the same poses. When two frames are identified to show the same pose and two markers have similar coordinates, it is very likely that they correspond to the same physical marker. The pose, however, is subjected to global rotation and translation, which has to be removed before the positional data of markers can be compared.

To accomplish this task [70] introduces a data structure they call Marker Set (MS). Each of the n virtual markers, for which an identity is found, has its own MS. A MS contains the data of all the frames for which the virtual marker exists and thus holds all information available about it as well as the relationships to other markers. We will use a nomenclature consistent with [70] in this section, thus naming the MS of the ith marker p_i. These n MS are now to be

grouped together into the n' physical markers. n' is assumed as the maximum number of markers captured in a single frame. So, if no frame exists, that contains all markers, the algorithm will evidently fail to deliver a correct result.

To determine which MS should be grouped and which shouldn't a distance function is created. This distance function allows us later to use a clustering algorithm on the MS. This method is based on the idea that the user repeatedly assumes the same global pose. The distance D_{ij} between a MS p_i and a MS p_j according to [70] is defined as the minimum distance between markers i and j over all pairs of poses, that is

$$D_{ij} = \min_{a \in p_i, b \in p_j} \|m_{i,a} - A m_{j,b}\| \tag{3.1}$$

Here a and b denote single frames of MS p_i and p_j. $m_{i,a}$ is the position of marker i in frame a, while A is the matrix that performs the global rotation and translation necessary to align the pose in frame b with that in frame a. So, for all frames of p_i, the distance to all frames of p_j is calculated one frame at each side at a time. Then the minimum of those distances is considered the distance between the MS.

To be able to calculate these distance values one assumption has to be made. It is crucial that the MS have at least three markers in common, of which the identity is already known. Otherwise A and thus the rotation and translation can not be estimated, which, however, should never happen anyway. If there are less than three markers, it would mean for the case of full body MoCap, that the tracker has lost more than 90 percent of the markers. That in turn would negatively affect the other stages as well, because they are depending on continuous data.

Since each MS has hundreds or even thousands of frames, it is evident that calculating the distance between all pairs of frames of two sets generates a gigantic computational overhead. Due to the fact that there are no huge changes of the pose in consecutive frames, especially at high capture rates, the calculations can be speed up significantly. Kirk et al. suggest using only sample frames from every MS. Starting from the sample frames that had the smallest distance, neighboring frames can be evaluated as well. Results show that this optimization produces the same matches as the original algorithm.

Furthermore, it has to be taken into account that MS which overlap can not belong to the same physical marker. Thus calculating the distance of MS that have frames in common is avoided setting it to the highest possible value.

Finally, having calculated the distance values of all pairs of MS, a clustering algorithm can be applied to group the sets together, producing n' markers. For that purpose Spectral Clustering is used, which is explained in more detail in [125].

Finding marker correspondence in imperfect data can be a challenging task and the methods described above failed in many of our data sets. Therefore, we have introduced numerous improvements to the algorithm as described in Section 4.2.1.

Other methods in related work for finding marker correspondence are based on rigid cliques of multiple markers placed on each segment [59], where inter-marker distances are used to identify markers. The latter, however, results in a large number of markers, because in approaches like [59] four or more markers are used per limb. Furthermore, some systems rely on markers being arranged in a predefined setup or manually assigned to skeleton segments.

3.2.1.2 Marker Clustering

After the 3D trajectories of all markers have been found, the next step is to determine which markers are placed on the same limb and group them together. For this purpose automatic methods for partitioning markers have been developed, like those suggested in [70, 136, 51]. These algorithms are all using measurements of the distance between markers to determine marker-limb associations. Again the method of [70] is described in more detail. The basic idea is, to assume that a limb of a human body is an (almost) rigid object. Thus, two markers placed on a single limb are never to move apart from or closer to each other, but always remain at a constant distance. In practice, however, the markers of one segment will move relative to each other due to skin or muscle movements, an ill-fitting body suit, or measurement errors of the tracking system. So the algorithm has to determine whether the movement comes from errors or - in case it is large enough - from the bend or twist motion of a joint. As a measurement, the standard deviations s_{ij} of distances between markers i and j are calculated over a representative sample of frames. Using the standard deviations, a distance or affinity matrix W is created, where the elements W_{ij} are depending on s_{ij}.

Based on this matrix markers can be clustered into groups each of which belongs to a segment of the skeleton. For the clustering algorithm Kirk et al. suggest Spectral Clustering.

3.2.1.3 Joint Position Calculation

After having clustered the markers into groups for every segment of the skeleton model, the topology and the joint positions can be estimated. These two tasks are very closely related; in fact the same method is used to infer them. To decide whether or not a joint is placed between two segments and, in case it is, to determine where to site it, a cost function is used. Two assumptions concerning the model are necessary for this cost function to work. First of all, human joints in the model are approximated by ball joints. This of course is a harsh idealization, especially for joints like the knee, where the center performs translational movements between one and two centimeters. However, in general, this approximation is close enough. The second assumption, which has to be made, is that markers stay at the same place relative to the bones. This, however, might not be entirely true either - skin movement and muscle deformation introduce variations - but the overall error is expected to be comparatively small.

Consider the case of only two segments A and B for which the assumptions apply. They are thus connected by an idealized rotational joint AB. A marker fixated on segment B of this structure can now be observed to always be at the same distance to AB. In fact it is moving around on a sphere centered at the joint position, when A is used as a fixed reference. Assuming that AB is unknown, this information can be used to estimate the joints position. An optimal joint between the two segments thus is placed on a position in each frame, where the distance to the markers of the adjacent segments remain the same over the whole capture time. In other words, the variance of distance over time between the joint and the markers has to be minimized in order to optimize the joint position. This is a non linear least squares problem and can be solved using a standard non linear optimization algorithm.

Using the variance as a cost function the topology of the skeleton model can be inferred. The value produced by the cost function for a (virtual) joint between two segments is called

joint cost. To obtain the topology the segments and joints are being interpreted as a graph, where the segments are the nodes and the joints are the edges. The joint costs are considered the edge weights. Therefore, the joint cost for all possible pairs of segments has to be calculated. The optimal skeleton then is the minimum spanning tree of the graph, which is inferred by the algorithm of Prim [133]. This gives us a hierarchical tree as representation of the skeleton model.

Minimizing the joint cost for every pair of segments, however, is computationally very costly, which is why [70] suggests an optimization for that matter. Instead of calculating the variance of distance over all frames only sample frames are used. Furthermore, decreasing the precision by which joint positions are optimized reduces computation time even more.

Joint Cost Function The core of all these calculations is the cost function, a sum of variances, which has to be minimized. Therefore, the function and its properties are described here in more detail. The variance is always evaluated between the markers of two clusters b_a and b_b, which in turn contain $|b_a|$ and $|b_b|$ markers. Only one joint c - between b_a and b_b - is considered at a time and its position in frame f is denoted c_f. Furthermore, the coordinates of c_f are xc_f, yc_f and zc_f, which are the variables of the cost function. Note that these coordinates have to be optimized for every frame in order to minimize the function. The position of marker i at frame f is given by $m_{i,f}$ and the number of frames for which a position of m_i exists $|m_i|$. $x_{i,f}$, $y_{i,f}$ and $z_{i,f}$ denote the coordinates of marker i at frame f.

Starting with the basic building block of the function, the average distance between a marker and the current joint is:

$$\bar{d}(c, m_i) = \frac{1}{|m_i|} \sum_{f=1}^{|m_i|} \sqrt{(xc_f - x_{i,f})^2 + (yc_f - y_{i,f})^2 + (zc_f - z_{i,f})^2} \tag{3.2}$$

The variance in distance of a marker and the current joint can be written as:

$$var(c, m_i) = \frac{1}{|m_i|} \sum_{f=1}^{|m_i|} \left(\sqrt{(xc_f - x_{i,f})^2 + (yc_f - y_{i,f})^2 + ...} - \bar{d}(c, m_i) \right)^2 \tag{3.3}$$

Finally, the joint cost is the sum of all the involved markers' variances divided by the number of markers:

$$jc(a, b) = \frac{1}{|b_a| + |b_b|} \sum_{m_i \in b_a \cup b_b} var(c, m_i) + \left[\alpha \cdot \bar{d}(c, m_i) \right] \tag{3.4}$$

The expression in brackets is suggested by [70] to avoid that the algorithm finds the trivial solution, which would be placing the joint infinitely far away. A joint at infinity has a variance of zero, as does the optimal solution. Adding a penalty to the joint cost, that increases with the distance to the markers, thus, avoids this problem. α here serves as a weight determining the importance of the additional term.

Now that the joint cost has been defined, the next step is to minimize it. Since the function depends nonlinearly on the parameters, the joint positions, sophisticated algorithms like Levenberg-Marquard or nonlinear gradient descent have to be used. According to Kirk, these two iterative algorithms have about the same performance, in both, speed and accuracy. However, they both depend on good initialization values to converge to a reasonable solution. Therefore, we have introduced the improvements described in Section 4.2.3.

3.2.1.4 Computation of Skeleton Parameters

After the joint positions have been optimized two joints connected to a single limb can still vary in distance. Thus the length of a limb is not fixed. This phenomenon can have a number of reasons, such as noise in the input data due to the inaccuracy of the tracking system, markers moving on the skin or the fact that human joints are only approximate rotational joints. However, it often is important to have a skeleton with rigid segments for performance animation and error measurement. For this reasons, a skeleton with rigid segments is created and then fitted. Therefore, as much useful marker and joint data as possible is collected over all frames. Then the sections of the skeleton are lined up individually and marker and joint data is averaged over multiple frames. These averages are then used in a second step to put the skeleton back together in a reference pose. The length parameters, that can be inferred from the newly assembled skeleton, are the best compromise between the measurements of all frames.

The algorithm is run on a per segment basis. For each segment only frames are used, which contain all the attached marker and joint data (i.e. have no occluded markers). The first frame for which this is true is considered the reference frame f_r. The set of marker/joint positions of the current segment in the ith frame f_i is denoted s_i. For all segments the following steps have to be processed

1. Iterate over all frames and find the set of segment-frame data $S = \{s_1, s_2 ... s_n\}$, where no markers of the current segment are occluded. Positions of markers and joints will be treated equally in the following and denoted with x'_{ji} for the jth marker/joint in s_i. Thus $s_i = \{x'_{1i}, x'_{2i}, ... x'_{mi}\}$, where m is the number of markers and joints of the segment. These positions are no absolute values but seen relative to the centroid, where the centroid of s_i is $\bar{c}_i = \frac{1}{m} \sum_{j=1}^{m} x_{ji}$, where x_{ji} are the absolute positions (where $x'_{ji} = x_{ji} - \bar{c}_i$).

2. The method of Horn [58] provides a closed-form solution to find the rotation and translation between two Cartesian coordinate systems based on the coordinates of a number of points. With this approach we calculate the transformation $T_i(s_i)$ that best aligns the segment s_i (i.e. its marker and joint positions x'_{ji}) with the position it has in the reference frame s_r. (thus $T_i(s_i) \approx s_r$). Note that connections between segments are broken up and

therefore after the transformation the skeleton for one joint has two different positions for both adjacent segments.

3. For every marker or joint now a cloud of positions C_j exists with $C_j = \{T_1(x'_{j1}), T_2(x'_{j2}), ..T_n(x'_{jn}))$

   The coordinates of C_j are averaged and produce rigid positions $\bar{X}'_j$ for each marker and joint attached to the segment.

After these steps have been completed for all segments a reference skeleton consisting of rigid bodies can be assembled. This can be used to measure parameters like segment lengths and can be deployed for example for animation.

Additionally, it can be used to improve the accuracy of joint positions by fitting it back to the marker data as described in our improved approach in Section 4.2.4 and evaluated using the error function introduced in Section 4.2.4.1.

Recent approaches published after our method, such as [6], introduced new algorithms for skeleton calibration working in real-time and using a Kalman filter to predict occluded markers. However, in the context of our work real-time performance is not an issue and marker tracking and prediction has shown to work robustly.

3.2.2 Skeleton Tracking

Skeleton tracking or fitting [136] describes the process of adapting the position and the joint rotations of the skeleton model in a way that fits the recorded data. Little information is published on the intrinsic functionality of skeleton tracking in commercially available full body MoCap systems. Some algorithms, like the global skeleton fitting described in [136], rely on the identity of the markers to be known. However, this is a constraint, which for reasons of stability, we don't want to depend on.

In an alternative approach, as used in [115], a rigid clique (or body) is placed on each skeletal segment. These cliques, consisting of three or more markers with fixed distances, can be used to derive position and orientation of the segment. As described in [59] these can also be trained dynamically. However, it depends on the markers to be fixated on rigid structures and four markers per segment, limiting applicability. Furthermore, kinematic models for different joints and body parts have been developed and analyzed (e.g. for the knee [4]), but little is published on generic models for the tracking of a human body.

CHAPTER 4

Full Body Motion Capture System

Medical indications requiring rehabilitation and foci of therapy can be very diverse, as pointed out in Section 2.9. For generic use in a rehabilitation context a Motion Capture (MoCap) system therefore has to provide movement parameters of different extremities, ideally capturing full body movements. Furthermore, Holden emphasizes the importance of skilled movements [57] for which rehabilitation applications need accurate tracking of limbs with multiple degrees of freedom. Therefore, full body MoCap, in this context, is defined as capturing the user's body in a way that allows for the reconstruction of six degrees of freedom (DOF) for every skeletal segment in a simplified skeleton model, which is available in a complete kinematic chain. MoCap systems based on optical tracking technology exist, but little has been published on the applied algorithms as described in Sections 2.4 and 3.2. Furthermore, state-of-the-art systems are expensive and often lack a convenient workflow needed for our field of application. Therefore, we have developed the MoCap system described in this chapter.

Our setup comprises of an infrared optical tracking system (see Section 3.1.1) and a number of software components performing different tasks in the MoCap and data processing pipeline. The physical setup can be seen in Figure 4.1. The theoretical foundations of the algorithms described here can be found in Chapter 3. For a more detailed description of the MoCap system's integration with applications and the rest of our framework please refer to Chapter 6.

Details of the system presented here have been published in [129] and [130]. In this chapter first an overview of the MoCap system's workflow in a rehabilitation scenario is presented in Section 4.1. Then Section 4.2 provides details on the methods used for the calibration of a skeleton model. Section 4.3 describes the algorithms used to fit the skeleton model to the tracked marker positions, thus providing real-time MoCap data. How this data can be pre-processed for use in an application is described in Section 4.3.3.

4.1 Workflow Overview

In order to successfully use a MoCap system for serious games in rehabilitation, it is important to have a workflow that can be easily handled. An overview of the workflow can be seen in

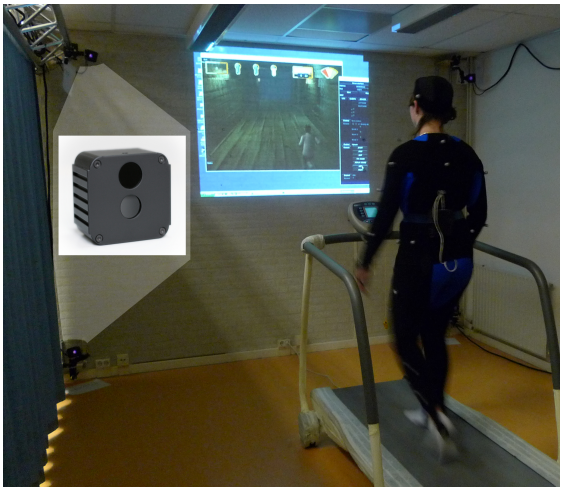


Figure 4.1: Our MoCap setup and serious game with magnifications of an iotracker-camera.

figure 4.2. The main steps are visualized and numbered in Figure 4.2 and listed below:

1. Skeleton calibration
2. Skeleton tracking/fitting
3. Data preprocessing and transformation
4. Game play and configuration
5. Feedback and evaluation.

Before the MoCap session can be started, the user puts on the motion suit equipped with passive markers as shown in Figure 4.1).

In the first step (1) a skeleton model has to be generated and adapted for each new patient. Using a separate tool, the "skeleton calibration assistant" (Figure 4.3), from the marker positions alone, a skeleton model is calibrated fully automatically using the algorithm described in Section 4.2. To execute the algorithm the user has to perform some initial calibration movements often referred to as the "Gym motion" [136, 54]. For a successful calibration the user moves the joints of interest to their full extent for about 10-20 seconds. For practical use with less experienced users, the skeleton calibration assistant shows movement sequences, which have proven to work well. However, no restrictions are imposed on the Gym motion except that it has to start in the T-Pose [44].

Out of the skeleton models generated from the sequences, the assistant automatically suggests the best for further processing to the user. The evaluation of the skeleton is based on the cost function described in Section 4.2.4.1. 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 and segments in the application, e.g. to an avatar. The labeled skeleton is then handed to the tracking software and the calibration assistant shuts down.

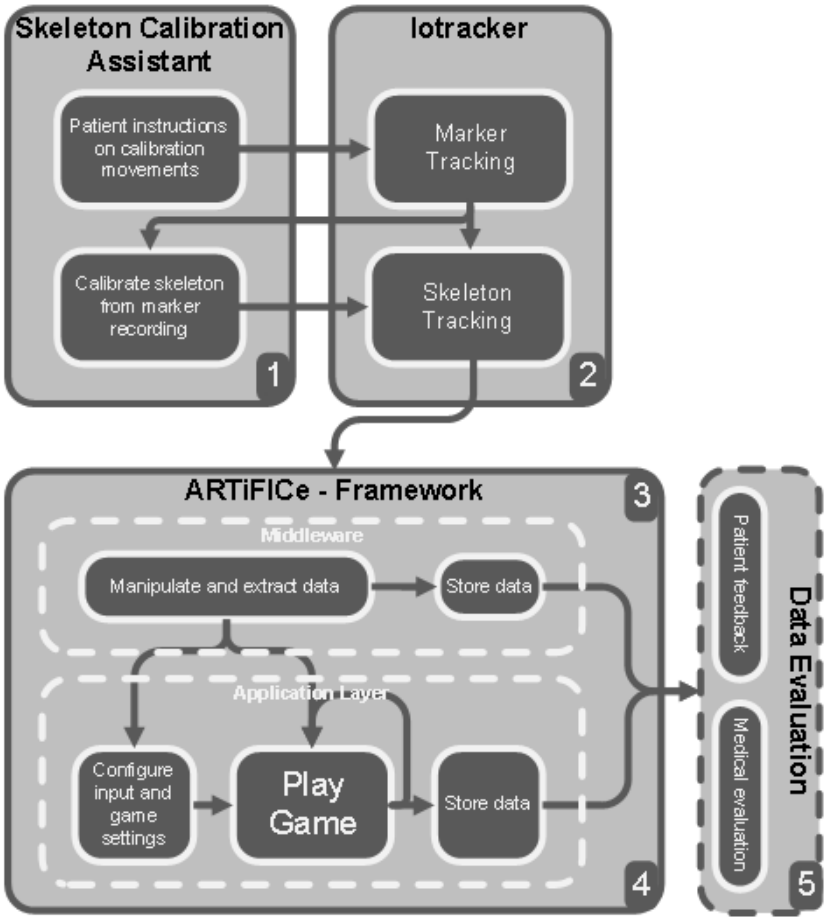


Figure 4.2: Visualization of the workflow in our system.

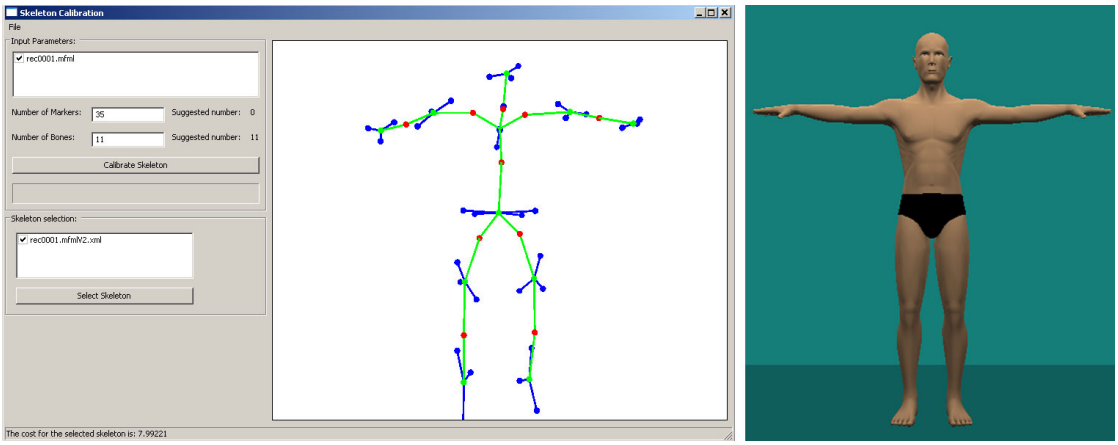


Figure 4.3: Skeleton calibration assistant: GUI and skeleton visualization (left), magnification of Gym motion instruction (right)

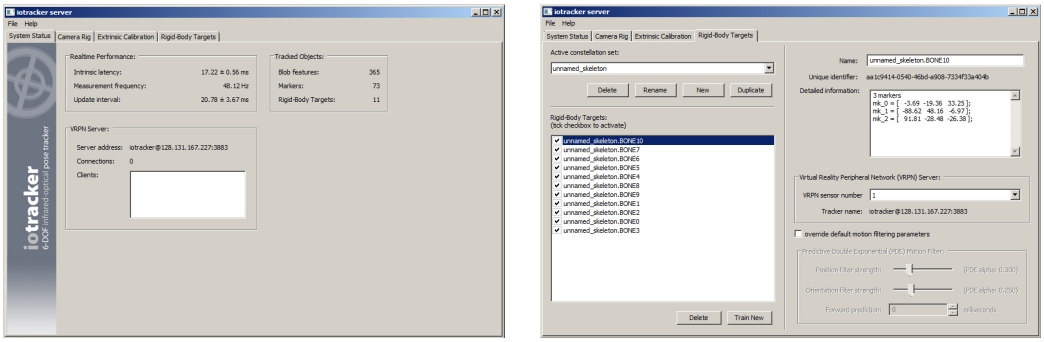


Figure 4.4: Motion capture module in the iotracker.

The MoCap module as shown in Figure 4.4, integrated within the iotracker server, is responsible for fitting the skeleton to the 3D positions of the markers in each frame (2). The algorithm is described in more detail in Section 4.3. The module loads the skeleton model file and optimizes the internal representation. Then it starts the MoCap process by fitting the model to the tracking data. The product of this step are angles for all joints of the skeleton model plus the position and orientation of the skeleton in 3D space. We consider this to be the "raw" MoCap data. The iotracker server implements a number of interfaces to external applications, which can retrieve the postural information from the MoCap module as detailed in Section 6.8.

For further use of input data often specific information (3) has to be retrieved (e.g. positions of certain limbs, movement velocities). To transform and extract this data from the raw data produced by the iotracker server and other devices described in Section 6.6, we are using OpenTracker as introduced in Section 2.8. Within OpenTracker we can easily manipulate and extract data from "raw" MoCap data using specialized nodes. Examples for extracted data used in the serious game described in Chapter 7 include:

1. Walking speed, which is calculated from the stridelength inferred from the heel-positions and the time stamps contained in the data-packages.

2. Reaching height, which is calculated from the position of a hand.

3. Turning velocity of the head, which is calculated from the rotation of the neck-joint and the time stamps.

How these parameters are calculated is described in Section 4.3.3. More information about OpenTracker and how it is used as middleware-layer of our ARTiFICe-framework, can be found in Section 6.5.1. In addition to live manipulation of the data, ARTiFICe is also used to write measured "raw" and extracted data to files for later evaluation by the therapist.

In the next step (4), the processed data is sent to the ARTiFICe application-layer, where it is used for interaction with the game. The game records game related data and also integrates a configuration scene, which allows customization for each patient, as described in more detail in Chapter 7.

Finally, the recorded data is evaluated to determine the progress of the patient throughout the game and in therapy (5).

4.2 Skeleton Calibration

The skeleton calibration algorithm allows quick adaptation of the system to different users, by automatically generating an approximate skeleton model. There is no need to take measurements by hand or place markers on precisely predefined positions, which are both time consuming matters.

A basic version of the skeleton calibration was already developed by the author within his master thesis [125] and therefore its workflow will be only briefly outlined here. For a full description the reader is referred to [125].

The skeleton calibration algorithm is flexible enough to cope with arbitrary articulated structures connected by rotational joints. This is especially interesting with a view towards universal accessibility. 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 and the skeleton calibrated for the required limbs. In the context of rehabilitation we can easily create skeleton models of body parts of interest by placing markers only there and running the algorithm. Thus, skeleton models for generic motion capture input, e.g. for a new rehabilitation exercise in a serious game, can easily be generated.

The applied algorithms follow the general approach introduced by Kirk et al. [70] and described in 3.2.1. However, several improvements were made, especially to make skeleton calibration more robust when using imperfect tracking data. Furthermore, mechanisms allow for more flexible marker placement and increased speed of the calibration calculations. The algorithmic improvements are outlined in the following sections and follow the same structure introduced in Section 3.2.1.

Data is recorded by the tracker during calibration sequences and processed in the following main steps.

4.2.1 Marker Correspondence

Passive markers in an infrared optical tracking system cannot be uniquely identified per se. The skeleton calibration, however, is depending on the markers being unambiguously labeled during the whole sequence. Therefore, in the first step Kirk et al. suggest identifying correspondence between markers from one frame to next based on the proximity of positions as described in Section 3.2.1.1. We use Double Exponential Smoothing Prediction (DESP) as introduced in Section 3.1.6 to estimate the position of a marker in the next frame(s), which better accounts for faster movements or lower frame rates.

We observed that the precision of the marker's 3D position-reconstruction might seriously suffer shortly before they disappear due to occlusions and after they reappear. This results in markers "jumping around", thus changing their position dramatically in consecutive frames. Since for our tracking system that doesn't occur too often, these problematic markers are simply discarded. Therefore, a threshold, which marks the maximum a marker is allowed to move from one frame to another, is used. Markers that "jump" over this threshold are then ignored for a short period of time. We use a threshold of 50 mm as well as a time delay of 1/20 of a second, which are depending on the tracking system used and have been determined experimentally.

Nevertheless, during longer occlusions even predicting marker ids is not always sufficient. In a second step we are therefore applying a new method based on the idea of Marker Sets described in 3.2.1.1 to match labeled sequences of markers with the support of clustering. Our new algorithm improves robustness of the labeling using the results of the second step (clustering). We define and minimize an intra cluster distance variation value in order to reduce negative influence of tracking errors (e.g. "ghost marker"). Also marker identification after occlusions is improved.

4.2.1.1 Using Inter-Marker Distances to Improve Marker Correspondence

Occlusions and erroneous data pose a challenge on finding marker correspondences, which with the methods proposed in 3.2.1.1 still fails for certain data sets. First we cluster the Marker Sets (MSs) as described in 3.2.1.1 to get an initial estimate of the MSs belonging to the same physical markers. This first estimate of marker correspondences is also clustered into segments as described in 3.2.1.2. In our novel approach, however, we then attempt to iteratively improve these correspondences and clusters by swapping and deleting MSs by following these steps:

1. Cluster the MSs into Merged Marker Sets (MMSs) using the minimum distance D_{ij} described in section 3.2.1.1. This is similar to the original step 1 except we consider only markers spatially local to the main marker of Marker Sets p_i and p_j for calculating the transformation matrix A between them. This way a user does not have to assume the same pose globally, but only with e.g. an arm. After clustering, an mms_i consists of a set $\{ms_1, ms_2, ...ms_n\}$ of MSs.

2. Cluster the MMS into Segment Merged Marker Sets (SMMSs) using the variance of distance $varDist_{ij}$. Note that $varDist_{ij}$ (see Section 3.2.1.2) is calculated from markers in the same frames (i.e. between two MMSs mms_i and mms_j, that have marker positions in at least two common frames. D_{ij} on the other hand is computed from Marker Sets that don't overlap in time as detailed in Section 3.2.1.2. This is analogue to step 2 in the original algorithm.

3. Iterate over the SMMSs and find the SMMS s_k with the largest $varDist_{ij}$ between two MMSs mms_i and mms_j. The algorithm then finds the MS responsible for the deviation in mms_i and mms_j and removes it to a pool P of unused MSs.

4. Try finding MMS(s) were one/(some) member(s) of P can be fit into so that $varDist_{ij}$ does not get significantly increased for all the pairs mms_i and mms_j in the corresponding SMMS. Alternatively look for pairs of MMSs mms_i and mms_j, which could be merged together.

5. If the overall $varDist$ is below a certain predefined threshold, or we have reached a specified maximum of iterations, stop. Otherwise goto 3.

Result of the algorithm is a set of MMSs, with each MMS corresponding to a physical marker and a set of SMMSs, with each SMMS corresponding to one segment. Furthermore, a pool of

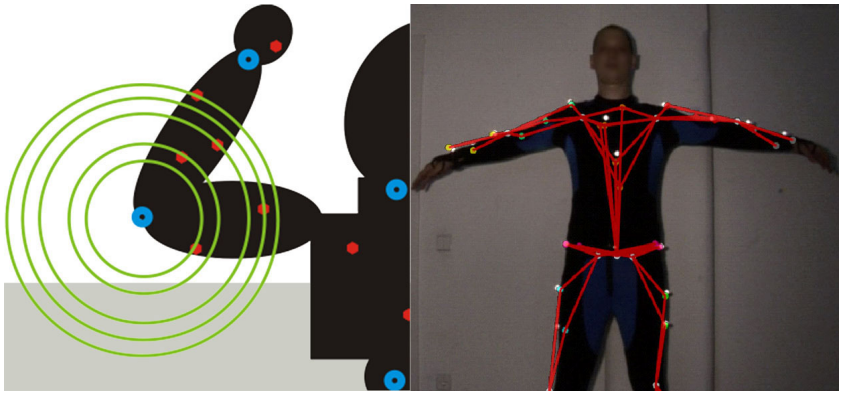


Figure 4.5: Visualization of the joint position optimization problem (left), Visual representation of a parameterized skeleton overlay with a video image (right).

unused MSs P contains marker data, which is thrown away. Depending on the size of the pool the user can be warned in case a lot of data is contained in the pool, which is an indicator, that there was erroneous tracking data (e.g. reflections in the environment or from retro-reflective elements on sports clothing interpreted as markers etc.)

While the effectiveness of the algorithm also depends on the number of markers on the segments, the two main advantages are:

- Detect if two MSs belonging to different segments are clustered together and correct it.
- Given there are two or more markers on a segment and at least one of the markers has an id. Then the $Dist_{ij}$ to that marker can help to identify other MSs on that segment. In other cases it might rule out that a MS belongs to a certain segment. Note, that the method described above does not help to avoid markers switching ids when on the same segment, if there is only two of them, because the value of $Dist_{ij}$ stays the same if they do.

4.2.2 Marker Clustering

The identification of markers situated on the same skeletal segment has been integrated with the search for marker correspondence described in the previous section and is therefore not detailed here. More information on the applied clustering algorithms can be found in Section 3.2.1.2 and [125].

4.2.3 Joint Position Calculation

Assuming that the joints are approximately rotational joints, the markers of two segments connected to a joint should ideally stay in the same distance to the joint during all frames (see Figure 4.5). As suggested in [136] and [70] this is formulated as a least square fit problem and optimized as detailed in 3.2.1.3. For this purpose we are using the nonlinear conjugate gradient method [134]. It minimizes a given nonlinear function using the function's gradient and details

on the implementation can be found in [125]. However, the success and performance of the optimization very much depends on the initial values. Since the joint positions are found iteratively, start values close to the optimum require far less iterations and avoid being trapped in local minima. For the estimation of the topology the center of gravity of two segments' markers is used as initialization for their in-between joint. Then the algorithm optimizes for a limited amount of about 20 frames and computes the joint positions/cost between all possible combinations of bones. The combinations with the lowest cost are then interpreted as the actual joints. These calculations already produce a first estimate of skeleton parameters such as offsets of markers from segments/joints.

In our extended method we have added an initialization step to the original algorithm, before the global optimization of the joint cost function described in 3.2.1.3 is started. Using the parameters from the calculation of the topology we can initialize the global optimization. Therefore, we use the first distance estimates in a local optimization algorithm to ensure good initialization for the precise calculation of the joints' positions.

4.2.3.1 The Local Optimization Function

The results of the topology estimation can be used to find good initialization values for the final joint calculation. This results in an improvement of performance and convergence as compared to the algorithm described in 3.2.1.3, where no specific initialization procedure is suggested. As an input our method uses the average distance $\bar{d}_{old}(c, m_i)$ between a joint c and the markers m_i, as it was found during the calculation of the topology. Then, for every frame (locally), the initial position of the joint is set to a value where the square of difference between the actual and the average distance is minimized. Again the nonlinear conjugate gradient method is used for the minimization of the cost function described here.

Since the optimization is performed for each frame individually and joint by joint only three parameters xc, yc and zc are needed for the joint coordinates. Before the local optimization, they are initialized to the center of gravity of the segments' $b'_a s$ and $b'_b s$ markers. The coordinates of the ith marker m_i at the frame to be treated are given by x_i,y_i and z_i.

$$ljc(a,b) = \frac{1}{|b_a| + |b_b|} \sum_{m_i \in b_a \cup b_b} \left(\sqrt{(xc - x_i)^2 + (yc - ...} - \bar{d}_{old}(c, m_i) \right)^2 \tag{4.1}$$

Minimizing this function gives us optimized initialization values for the joint position in that frame described by xc, yc and zc. This method has shown to be stable and robust, especially for erroneous data. Calculating initial values in a closed-form approach using triangulation has also been tested, but has shown to be less robust to inaccuracies of tracking or the initial parameter estimation.

4.2.4 Computation of Skeletal Parameters

Once the joint positions are calculated for every frame, the offsets between markers and joints can be calculated and averaged as introduced in Section 3.2.1.4 and detailed in [125]. These

offsets can then be used in skeleton tracking to identify markers and correctly position segments and joints in the marker point cloud. Figure 4.5 shows a visualization of the skeleton calibration results. In our workflow we want to (help the user) decide, if the calibration result is sufficient. Therefore, we need an error estimate of the skeleton model.

4.2.4.1 Error Estimation of Skeleton Parameterization

After having calculated parameters of the skeleton model consisting of rigid sections, it is important to assess the quality of the model and if it can be used for tracking. With a view towards the application scenario in an every day clinical setting, we want to do that with as little user intervention as possible. We therefore introduce an error measurement based on the joint cost function described in Section 3.2.1.3. It produces a single value, which can be easily thresholded or interpreted otherwise. The idea is to assess how well the skeleton model fits the data, therefore, its pose has to be adapted for each frame, so it matches the originally captured markers of the calibration sequence. Starting at the root segment, the joint transformations are adapted successively until all marker positions have been fitted optimally.

The skeleton is fitted to the data one segment at a time, which allows a closed-form solution. This in turn has the advantage of being very fast and robust, while the result still shows descriptive.

The Algorithm From the standard algorithm described in 3.2.1.4 we retrieve a skeleton model, that assumes the pose in the reference frame r. This skeleton model consists of multiple segments $sr_{i,r}$ at different levels in the skeleton hierarchy l. Each of the segments has a number of markers $\{m_{1,r}, m_{2,r}, ...m_{n,r}\}$, the positions of which are defined relative to the inner joint of the segment. Now joint transformations have to be found to fit the skeleton model to the marker positions in a frame f.

Let $T_{i,f}$ be the transformation matrix containing the rotation of segment i from the reference frame to the desired pose in frame f. For each frame f we now want to find the transformations necessary to best fit the markers $\{m_{1,f}, m_{2,f}, ...\}$ at the frame.

First the root segment $sr_{0,r} = \{m_{1,r}, m_{2,r}, ...m_{n,r}\}$ of our parameterized skeleton in reference frame r has to be matched with the markers of the root segment $s_{0,f} = \{m_{1,f}, m_{2,f}, ...m_{n,f}\}$ at frame f (where hierarchy level $l = 0$). In other words the transformation $T_{0,f}$ has to be found, where $s_{0,f} \approx srt_{0,f} = T_{0,f} * s_{0,r}$. Again the method of Horn [58] as described in Section 3.2.1.4 is used for a closed-form solution of the transformation. The transformation is applied to the whole skeleton structure including all children.

In the next step we calculate the transformations at hierarchy level $l = 1$, i.e. for all children segments of the root. Similarly to the previous step the segment of the reference frame with its markers $sr_{i,r}$ is matched to the corresponding segment $s_{i,f}$ in frame f calculating the transformation $T_{i,f}$. The transformation is again applied to all children of the segment. Calculation of transformations are continued until the final level of the skeleton hierarchy has been reached. Finally, from the transformations in the skeleton hierarchy, we can calculate transformed global positions for all markers contained in the skeleton model for all frames denoted as $mrt_{f,j}$.

The error function measures the average square of distances between the marker positions ($m_{f,j}$ with coordinates $x_{f,j}$, $y_{f,j}$ and $z_{f,j}$ in frame f) as measured by the tracking system and the markers of the reference skeleton transformed into the pose of frame f ($mrt_{f,j}$ with coordinates $xrt_{f,j}$, $yrt_{f,j}$ and $zrt_{f,j}$) over all frames. The function is thus given by:

$$error = \frac{1}{|f|}\sum_{i=1}^{|f|}\left(\frac{1}{|m|}\sum_{j=1}^{|m|}\left((xrt_{i,j}-x_{i,j})^2+(yrt_{i,j}-y_{i,j})^2+(zrt_{i,j}-z_{i,j})^2\right)\right) \tag{4.2}$$

Results of this function allow to assess, compare and if necessary discard calibrated skeleton models. Typical values for the error are below 100 in our system with the marker coordinates given in millimeter.

Skeleton calibration is an offline procedure during which skeleton models are calculated sequentially from one or multiple calibration sequences. Due to the offline nature of this processing step, sophisticated algorithms such as non linear optimization or spectral clustering can be used. However, the MoCap process itself is performed in real time.

4.3 Skeleton Tracking

The skeleton tracking algorithm detailed in this section attempts to fit the skeleton model to the marker positions generated by the iotracker. The methods are partially extensions of those designed for target tracking described in Section 3.1. The basic steps will therefore only be referenced in this section.

Using the skeleton calibration described in the previous section, we are able to reconstruct a full body kinematic constraint model of the user's skeleton in an offline process. This kinematic model is stored in an XML file as hierarchical chain of rigid segments, connected by rotational joints as illustrated in Figure 4.6. Every segment is associated with one or more optical markers, the position of which relative to the segment remains static, building a rigid clique.

At runtime the iotracker produces 3D marker positions as described in Sections 3.1.2 and 3.1.3. The skeleton model is then fitted to these marker positions to obtain a skeleton pose for every frame.

The full parameterization of the skeleton model (body posture) is given by the set of all joint angles plus the 6 DOF global transformation of the kinematic tree (usually the upper torso). During the calibration procedure our tools automatically identify the upper torso (or any other segment of choice configurable through the skeleton XML file) and label it as the root segment.

Our skeleton model fitting algorithm is composed of the following steps (see Figure 4.7). Algorithmic details are given in the next subsections.

1. **Prediction of the expected body posture.**

   Upon first run or re-initialization no previous body posture estimates exist. Therefore, in these cases we are using a generic one (a standing pose or the so-called T-pose) as

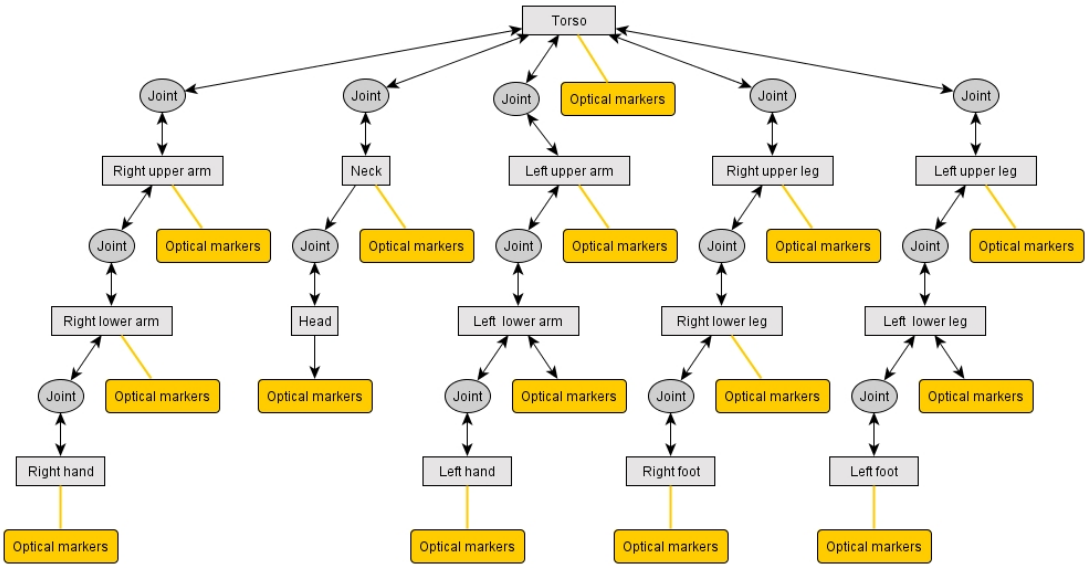


Figure 4.6: Kinematic model for skeleton tracking.

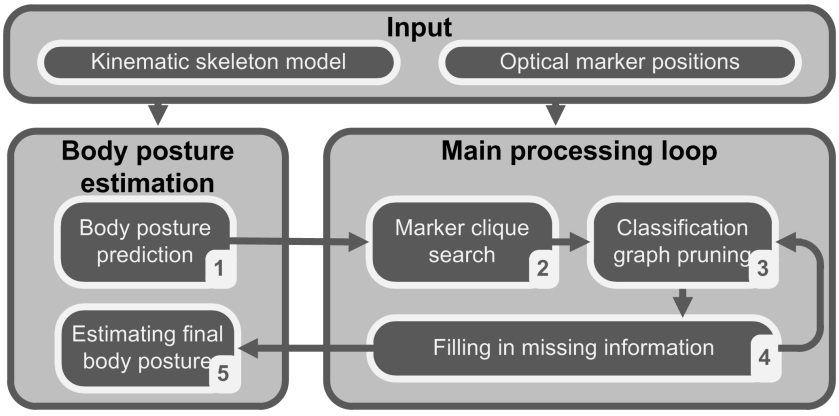


Figure 4.7: Overview of the skeleton tracking pipeline

initialization. Otherwise, we use predictive forward kinematics to compute the expected body posture from the last known pose as described in Subsection 4.3.1.

2. **Rigid marker clique search, generation of a classification graph.**

   We try to identify every segment, with a rigid clique of two or more optical markers, in the point cloud, by using inter marker distances. The method used here is similar to the one described in Section 3.1.5. However, we usually have less markers on a segment than on the targets described in Section 3.1.1. Therefore, most of the time, clique search will result in multiple hypotheses how the model could be fitted to marker positions in the current frame. From the differences between the actual relative distances and the ones stored in the skeleton model, a model alignment error is calculated.

   Every hypothesis, whose model alignment error lies below a certain threshold, is stored in the vertex of a classification graph, instead of immediately deciding on a unique solution. The graph edges represent geometric transformations between different cliques and are computed upon insertion of new vertices. Details on this step are described in Subsection 4.3.2.

3. **Classification graph pruning.**

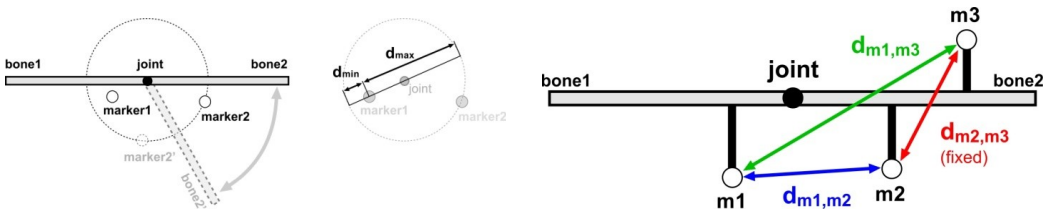
   From the hypotheses in the classification graph, we first compute the absolute pose of the associated segment in as many degrees of freedom as possible. We then test this pose against the expected posture (determined in Step 1). If the difference between both poses exceeds a certain experimentally determined threshold (12 degrees and 50 millimeters in our implementation), 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 skeleton model by pair wise examination of adjacent segments. When we find an edge that does not satisfy the kinematic constraints, we remove both connected vertices from the classification graph.

4. **Filling 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 the middle joint pose by an iterative inverse kinematics approach [24]. From here, we repeat Step 3 and 4 until all ambiguities are resolved.

5. **Posture estimation.**

   In most cases, all ambiguities are resolved in Step 3 and only one vertex remains for each clique (containing the hypothesis, how it best matches the data). In the rare event that there are multiple hypotheses remaining, we assign scores to the hypotheses based on the model alignment error. We then select the highest-scoring hypothesis from the classification graph and remove the remaining hypotheses. We label all markers according to the clique and associated segment they are matched with. If necessary, occluded optical markers are substituted with “virtual markers”. Based on the uniquely labeled optical marker set, we iteratively compute the full skeleton parameterization (using forward kinematics) with the



(a) Distance limits for markers on adjacent segments.

(b) Pair-wise distances of multiple markers on adjacent segments.

Figure 4.8: Distance limits of markers on adjacent segments.

goal of minimizing the model-alignment error, which is similarily calculated as described in Section 4.2.4.1.

4.3.1 Prediction of the Expected Body Posture

The full parameterization of the skeleton model's transformations is given by the set of all joint angles plus the 6-DOF transformation (position and orientation) of the root segment. 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 segment of choice) and label it as the root segment, which will be named segment 0 in the following. For a kinematic chain of n segments, let θ_i be the joint-angle between segment i and segment $i-1$. Given $\theta_1 ... \theta_n$, the frame of segment n relative to segment 0 is:

$$T_n^0 = \prod_{i=1}^{n} T_i^{i-1}(\theta_i) \tag{4.3}$$

Where $T_i^{i-1}(\theta_i)$ is the transformation matrix from the frame of segment i to segment $i-1$. In absence of a valid measurement for $\theta_k (k = 1..n)$, it is not possible to compute $T_i^{i-1}(\theta_i)$ for any $i \geq k$. We solve this problem by employing a double-exponential smoothing-based prediction filter (DESP), as described in Section 3.1.6, for every joint angle θ_i, which provides us with estimates of the joint angle in absence of actual measurements. DESP can be used to predict the rotation (in quaternion representation) multiple frames to compensate for short occlusions and other tracking errors, but the quality will naturally decrease with the number of frames, for which no actual measurements are available.

4.3.2 Algorithmic Extensions for Kinematic Chains of Rigid Cliques

In an unstructured point cloud, unique marker identities can only be established by inspecting known pair-wise distances between markers. For carefully designed rigid-body targets the algorithm for clique search described in 3.1.4 works very well. However, with the proposed arbitrary positioning of markers on a MoCap suit, 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 segment in order to establish unique marker labels. Therefore, we extended the complexity reduction algorithm described in Section 3.1.4.1 from a single rigid clique of markers to kinematic constraint models. We precompute additional multi-dimensional marker-to-marker distance lookup tables from the calibrated skeleton. This computation is performed as a separate offline step. Figure 4.8a) shows a simple two-segment skeleton fragment connected by a spherical joint, where each segment 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 segments.

Distance lookup-tables can be extended to span multiple markers on adjacent segments as pointed out in the following example.

Example:

For any given distance d_{m_1,m_2} (where $d_{min} <= d_{m_1,m_2} <= d_{max}$) between marker m_1 (on $segment1$) and marker $m2$ (on $segment2$), distance d_{m_1,m_3} between marker $m1$ and marker $m3$ (on $segment2$) can be stored in a lookup table. Figure 4.8b) illustrates this example. For reasons of efficiency and memory usage, distance values are discretized with step sizes of multiple millimeters. Using the distance lookup tables described above, we iteratively inspect the distance relationships between markers on adjacent segments. This allows for significant improvements in marker labeling accuracy, with minimal computational overhead.

The output of the skeleton tracking step is the body posture as described above for every frame and so-called "points of interest" (POIs) as described in the next section.

4.3.3 Points of Interest

A POI is a specific position defined within the skeleton model. It can be defined relatively to a joint while being located on a segment. POIs can define positions on the skeleton that have a special relevance for (medical) applications. 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 the (rehabilitation) application (e.g. heel position for gait-analysis or a point on the upper position of the shoulder to detect shoulder-elevation).

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 segment 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 segment. This position can be easily specified and calculated as soon as the tracking system has computed the pose of the skeleton from the real world markers. Furthermore, the

tracking system 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. in [144]). 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, segments or other relevant landmarks.

An example, where two physical marker sets (Figure 4.9) have been mapped to the same virtual marker set, is shown in Figure 4.10. The visualization application is based on the tool developed in [45].

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 chronic pain rehabilitation game and the exercises (see Chapter 7), 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 thus measure the elevation-level of the arm.

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 segment to define the order in which it is processed within the skeleton hierarchy. From this relative position the absolute position is calculated for every frame by the tracking-software.

For computation the hierarchy of the skeleton is traversed and the segment/joint-positions updated according to the current joint-angles. Finally, the offset of the POI is added to the referenced joint/segment-position resulting in the absolute POI-position. The XML-representation of the POI and its translation is then transmitted through the interfaces described in Chapter 6.

4.3.3.1 Movement Velocities

From tracking data of different POIs over time walking speed, head rotation velocity and hand movement velocity are calculated. The *IOTMCSkelMod* OpenTracker node (see Sections 2.8 and 6.6.3 for details) uses the heel positions for calculation of the walking speed on a treadmill. The algorithm expects the vector of the walking direction as input (usually defined according to the placement of the treadmill in the room). Once events with heel positions start arriving they

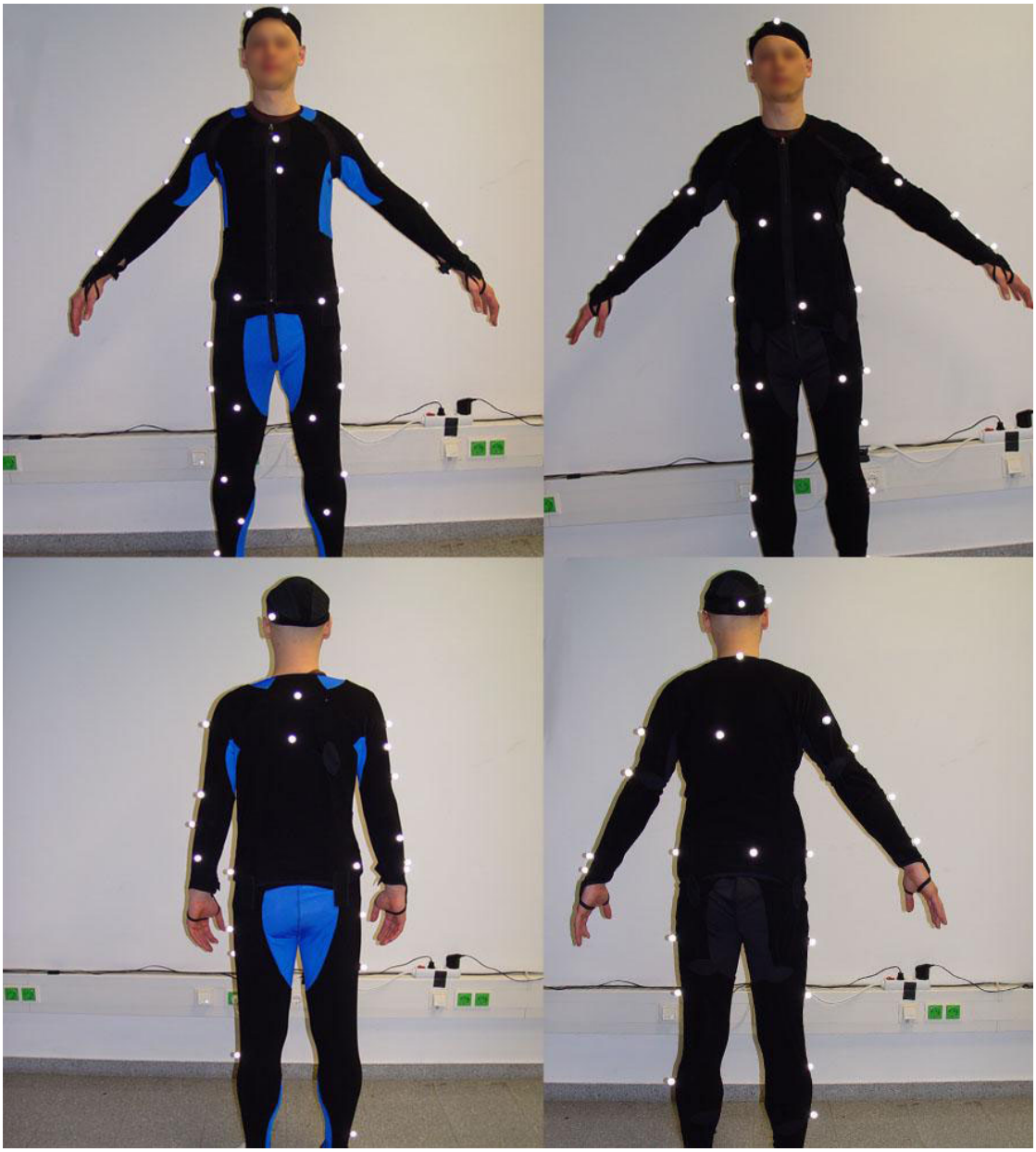


Figure 4.9: Two MoCap-suits with different marker-sets (passive markers).

are cached and analyzed for local maximum and minimum positions in the direction of walking. Once an extreme position has been recorded for each foot the stride length of the corresponding step can be calculated. From the input's timestamps we calculate according step times and subsequently velocity of each step. The overall walking velocity transmitted to the application layer is averaged over several steps to increase robustness. Furthermore, error detection mechanisms ensure that only regular steps are considered and issues like temporary occlusion of markers or

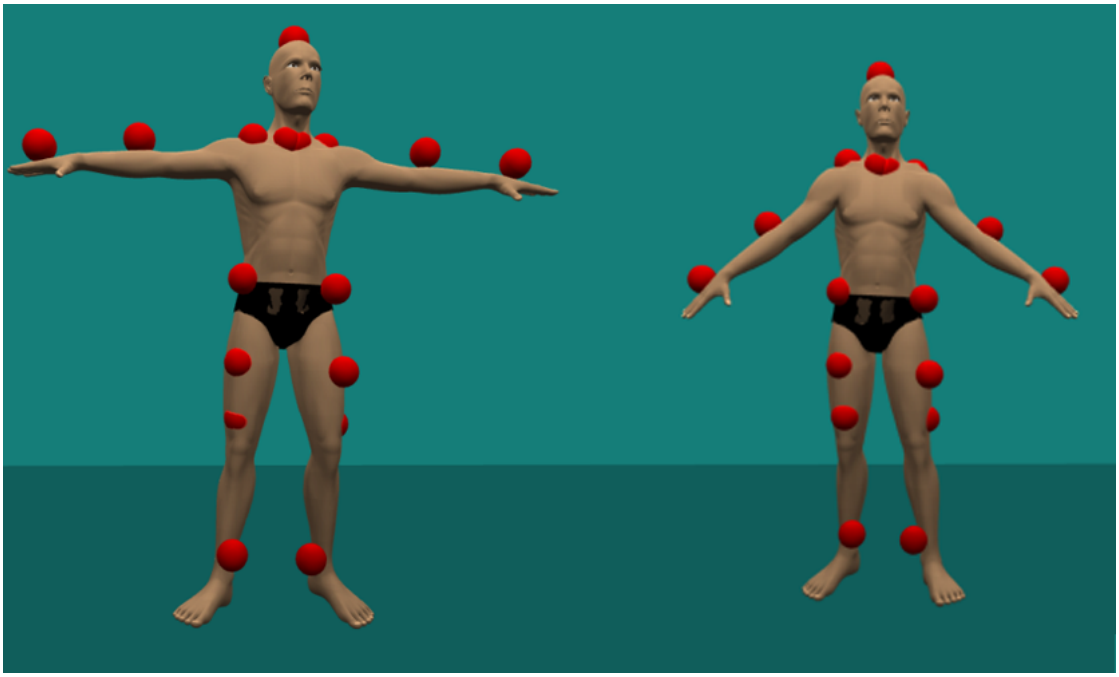


Figure 4.10: Points-of-interest visualized. Please note that the avatar is just added to better illustrate the locations of the POIs and not perfectly aligned with the real-world-position.

other tracking failures don't produce faulty velocity measurements. This is especially important for the application described in Chapter 7, where we want to measure walking velocities with an accuracy of 0.1 km/h, while we expect approximately constant stride lengths.

Hand movement velocity is calculated based on POIs located on the wrist and their timestamps. We first measure distances between successive positions and calculate the velocity based on the time differences.

Rotation speed of the head is measured similarly, only that it is calculated for the different rotational axes separately, thus resulting in three different velocity measurements.

4.4 Technical Evaluation

4.4.1 Performance

The iotracker server application runs on a quad-core CPU (Intel Core2Quad Q9300 with 2,5 GHz with around 75% CPU load). The skeleton tracking is implemented in a highly parallelized way using OpenMP and utilizes all cores. Especially step 2, which is the most computationally intensive, is well suited for parallelization. During our evaluations we were running the iotracker server and the game (developed in Unity3D) along with other input modalities (automatic speech recognition and electromyography) on the same PC. Therefore, we used a Workstation with two Intel Xeon DP X5355 processors (a total of 8 cores). CPU load was at around 85% during active

gameplay. The latency of the MoCap system consists of Firewire data transfer latency of 18ms (in case of 60 Hz tracking), since all camera images are transferred to the PC for processing. Additionally, overall latency is determined by the time needed for skeleton tracking, which is typically between 5 and 10 ms for one skeleton on the above CPUs.

4.4.2 Infrared-Optical Tracking System

Basic measurements of the iotracker have been presented in [105]. Measurements indicate low latency ($20-40ms$) for the targeted amount of markers, minimal jitter (RMS less than $0.05mm$), submillimeter location resolution and relatively high accuracy with a RMS distance error of $\pm 0.5cm$. For a more detailed description of the marker tracking performance please refer to [105].

4.4.3 Skeleton Calibration

In the standard skeleton configurations, that have finally been used in the application described in Chapter 7, the upper body is connected to the upper arm directly via a rotational joint. However, in early experiments we placed additional markers on the shoulder blade and calibrated it as an additional segment in the model.

An evaluation of the serious game with ten patients has been conducted as described in more detail in Section 7.5. Each patient was participating in the game interventions in a period of four weeks with one or two game sessions per week. During the evaluation the usability of the skeleton calibration procedure was rated good. All patients were capable to perform the requested skeleton calibration exercise and the calibration assistant produced and selected a functional skeleton model for each individual in almost all cases (87.5 %, see Section 7.4.2 for more details). Since accurate skeleton calibration depends on a correct labeling step, it can be assumed that labeling also worked in at least 87.5 % of the calibration sequences.

Furthermore, the MoCap system was working stable and produced robust tracking data throughout the evaluation. Due to the lack of ground truth no absolute measurements of the system's accuracy have been made. Measuring human motion data with non-intrusive means (e.g. a goniometer) is usually highly inaccurate and depends on the tester and articulation used. Acquiring kinematic data of a human body by intrusive means on the other hand requires an enormous effort (e.g. radiostereometry [150] or bone pins [4]) for which we lack the means.

4.4.4 Skeleton Tracking

Visual inspection and subjective evaluation of the motion data showed smooth movements with little jitter. Evaluation of the joint angles showed also little jitter and high relative accuracy. Plots of unfiltered tracking data produced by our system can be seen in Figure 4.11 (curves of angles with graduation in the sub-degree domain). We have measured the rotation of the head relative to the torso on the up-axis. Even during fast motions like shaking of the head our system produced a smooth curve of measurements and even the inferred rotation-velocity introduced little jitter. When looking at slower motions our system produces values with sub-angular accuracy, which exhibit high plausibility. In addition, movement parameters calculated

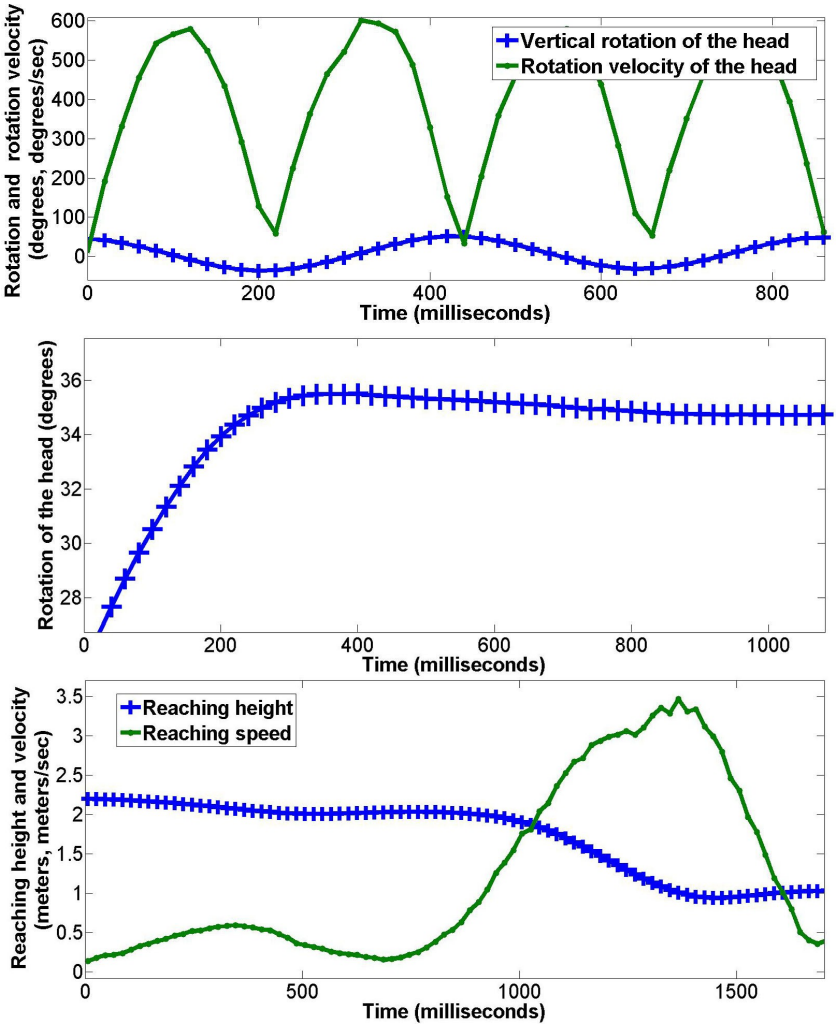


Figure 4.11: Rotation angle and velocity of the head relative to the torso around the up-axis during fast movement, i.e. head shaking (top), rotation of the head in the sub degree domain (middle), reaching height and velocity of a hand (bottom)

from positional information show little jitter in position and velocity. Finally, measurements regarding range of motion of assessed exercises show good repeatability.

CHAPTER 5

Consumer Devices as Alternative Motion Capture Devices

5.1 Motivation

As stated in Chapter 1 home-based exercises play a crucial role in rehabilitation. The iotracker, however, is too expensive and difficult to setup in a home environment and we have therefore evaluated Microsoft Kinect (Section 5.2) as an alternative MoCap solution to iotracker in Section 5.3. This low-cost and plug-and-play tracking component can capture certain movements and exercises with an accuracy, which is sufficient for our rehabilitation context, while it might not be so well suited for others. However, a single Kinect only covers a relatively small tracking space. Therefore, we have built a system consisting of several Kinects to assess the applicability in wide-area tracking scenarios as detailed in Section 5.4. The systems and findings described in this chapter have been published in [131, 127] and [93].

5.2 Setup

We have integrated Microsoft's Kinect in our ARTiFICe framework [93] as described in Chapter 6. Kinect has been designed as a natural user interface using gestures instead of controllers to provide input to conventional games. As opposed to the iotracker based MoCap system, it uses only one camera for tracking and works without markers. Instead, a dot pattern is projected onto the environment and user(s) by an infrared laser. Projected points are traced by the camera. Projector and camera are positioned at a certain calibrated distance from each other within the Kinect casing. Therefore, the dots recorded in the camera image and the original pattern can be used to reconstruct a depth image. The depth image in turn is used to calibrate and fit a human skeleton model, resulting in a skeleton pose [135]. In addition, from skeleton poses over time gestures can be recognized using the middleware layer "Flexible Action and Articulated Skeleton Toolkit" (FAAST) [143]. In addition to the skeleton pose, FAAST recognizes gestures, that

can be used within a game. FAAST wraps the OpenNI/NiTE framework as well as Microsoft's Kinect for Windows SDK, that provide alternative solutions for lower level access to the Kinect. The accuracy of Kinect is limited by the resolution of the camera and by the fact that only one perspective is used.

5.3 Evaluating Accuracy of a Single Kinect

We conducted a brief technical evaluation of Microsoft's Kinect. The main focus of this evaluation was to determine, whether certain exercise movements (as they are used for the serious game described in Chapter 7) can be tracked satisfactory. To compare our MoCap system with Kinect we have recorded unfiltered movement data simultaneously with both systems. We ran iotracker and Kinect on two different workstations connected by a local area network. The data from Kinect was streamed to the iotracker PC, where time-stamped values were written to file. For movements typically used in the mini games we then compared significant parameters between the two MoCap systems. In addition, we test played two of the mini games described in Chapter 7 by using Kinect. Although both systems work within the infrared spectrum of light, the systems did not negatively affect each other's measurements. The first aspect, which we investigated was the user's hand positions and parameters derived from these. We started comparing the reaching height relative to the torso, as reported by both systems. Figure 5.1 shows the height values of the right arm as they are used within the game. Measurements show good conformance except that the hand position calculated by Kinect extends slightly larger (about 5 to 7 cm). This is amplified by the fact that the hand position is defined slightly different in both systems. Later studies like [97] confirmed deviations of around 7 cm for the wrist positions. Furthermore, 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. Figure 5.1 shows a comparison of velocity graphs. Since velocity is being used to determine smoothness of motion this is considered problematic for our application.

Due to the positional differences, latency between the systems could not be exactly determined. However, aligning the extreme positions of the movements we have estimated the Kinect data to arrive somewhat below 100 milliseconds later than the iotracker data. Taking into account the latency of 20-40 ms of the iotracker and that the abovementioned network transmission between the two workstations running iotracker and Kinect adds a few milliseconds, overall latency of Kinect is estimated to be around 100ms.

The second aspect under investigation were walking movements. We have taken a look at variations of foot positions in the walking direction, while walking on a treadmill, which are used to calculate the stride length and walking speed consecutively. The Kinect sensor was placed in front of the treadmill, creating a difficult scenario in which the player was partly occluded by the treadmill's structure and foot positions could only be calculated from depth information. A visualization of the setup's depth image provided by FAAST can be seen in Figure 5.1. Once Kinect properly recognized the player within the structure, it performed rather well for our purpose. Iotracker reported larger values for stride length compared to Kinect. That could be caused by different definitions of foot positions (a comparison of foot positions is shown in

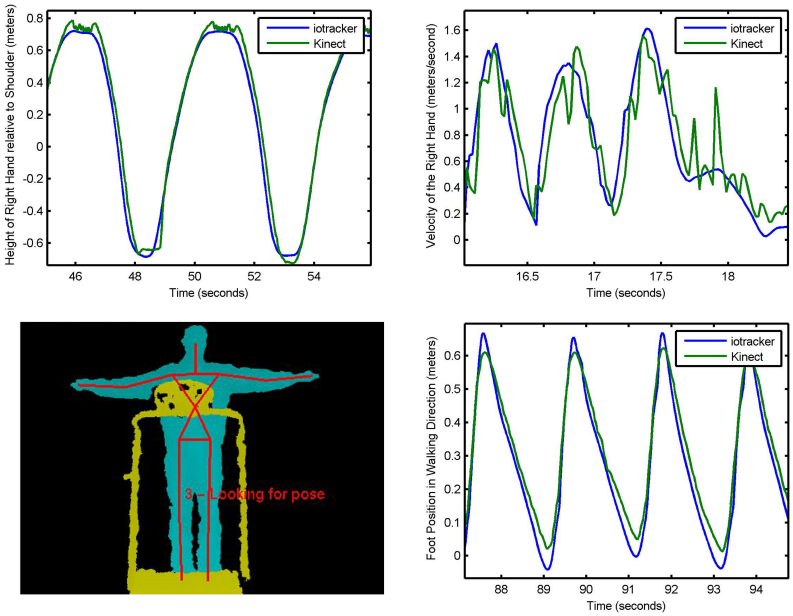


Figure 5.1: Comparison of measurement between iotracker and Kinect/FAAST. Handposition and velocity (upper row). Treadmill and test user with skeleton model visualized by FAAST during the walking measurements (lower row left) and foot positions in walking direction (lower row right)

Figure 5.1). Jitter is again recognized to be stronger than in our system and even stronger than for the hand positions. Nevertheless, Kinect provides useful measurements for our purpose.

Head rotations are another aspect that is crucial for the serious game described in Chapter 7. However, tracking of head rotations was not available in OpenNI/NiTE. This feature was only introduced with Microsoft's Kinect for Windows SDK version 1.5, that was released approximately one year after we ran these tests. Therefore, we could not evaluate these measurements nor test that specific part of the game. The head tracking solution of the Kinect for Windows SDK is based on face tracking from an RGB image. Therefore, technical limitations of this approach are very likely to lead to relatively inaccurate measurements.

Overall Kinect works surprisingly well as an alternative MoCap system used to control our game targeted at rehabilitation of chronic pain patients. Our tests have shown that it correctly captures some of the exercises used within the game. Two out of three mini games were playable, although with certain restrictions. Kinect, however, can't measure all required parameters and lacks accuracy required for others, especially with regard to a medical evaluation.

5.4 Extending Tracking to Wide Area

As introduced in Section 2.5, recent developments in consumer hardware have introduced markerless MoCap as input to games for entertainment but also serious games for rehabilitation [79].

Low-cost depth cameras enable applications to capture human movements in a non-intrusive and stable way. Ready-to-use software modules and middleware, for retrieving sensor data and human pose information of a simplified skeleton model, have been distributed by sensor manufacturers [91, 110]. Furthermore, application interfaces make integration in existing and customized applications possible. Finally, ease of deployment and good usability makes them applicable in a home environment. By moving control into the third dimension compelling experiences can be created and new ways of interaction established.

To a limited extent this has been explored in different ubiquitous computing projects, where boundaries between devices (ideally) become transparent to the user and a unified form of interaction can be applied (for an overview see [107]). However, the application of wide area tracking in such a context provides a whole set of new possibilities: As introduced in [162] MoCap data can be used to interact with three-dimensional objects or projected surfaces and menus. In a game a virtual avatar could follow a player around in his apartment and could be interacted with by gestures. If the game is distributed on different platforms, a PC, game console or smartphone could then provide visualization of the game content. Serious games for rehabilitation could also greatly benefit from an extended capture space. Only then for example natural walking and gait analysis could be integrated in a game.

Nevertheless, systems, such as Microsoft's Kinect, are designed for use within a classical home entertainment setup and are therefore limited to confined spaces. The tracking volume is rather small due to the limited field of view of the integrated camera and technical constraints (i.e. range of IR illumination) restrict the possible distance between user and sensor. In addition, occlusions, especially with multiple users, result in incomplete postural information. To our knowledge, our system presented in the following, was the first to use multiple depth cameras to enhance full-body tracking data and expand the MoCap area beyond a single room. We are integrating commercially available depth cameras and motion capture middleware in a flexible networked software environment, which allows us to improve the tracking quality and scale the captured volume almost arbitrarily. Deploying multiple Kinects in a single room causes interference, if the projected dot patterns of the individual sensors overlap. However, a certain amount of overlap between sensors is necessary to allow seamless coverage of a capture volume and for calibration as described in the next subsection. We found that skeleton tracking quality did not deteriorate substantially even with two adjacently positioned Kinects. These finding has been confirmed by [11], which was published simultaneously to our work [127]. Berger et al. also measured the percentage of error pixels for multiple Kinects and different placements, which increased with the number of Kinects. However, the error largely stayed in the low one-figured range except for highly specular materials and four Kinects. Nevertheless, we have tried to limit overlap to smaller areas and a small amount of sensors to reduce unnecessary errors. To reduce errors further or in situations where more overlapping sensors are required, an approach similar to [64] could be used. Kainz et al. attached vibration motors to the sensors to blur the pattern for other simultaneously measuring sensors.

5.4.1 Software and Calibration

We are using consumer hardware (i.e. Microsoft's Kinect [91] or Asus Xtion [7]) together with the OpenNI [110] software framework and NiTE [109] middleware. OpenNI offers interfaces for

low-level communication with devices, while NiTE offers higher-level features such as skeleton tracking and gesture recognition. For reasons of simplicity we will call the hardware components "sensors" in the rest of the chapter. In order to use data from multiple sensors, it has to be transformed into the same coordinate system. For that, the relative positions of sensors to each other are needed. We utilize the MIP Multi Camera Calibration tool [122], which can be used to calibrate intrinsic and extrinsic parameters of RGB and depth cameras. The origin of our world coordinate system is that of an arbitrarily chosen master sensor. The other sensors' positions are calibrated relative to the master. Therefore, the sensors have to have an overlapping field of view. With the Kinects' RGB cameras we take multiple pictures of an A0-sized checkerboard calibration pattern in the overlapping regions. From the images we calculate the transformation from one camera space to the other using the calibration tool. For sensors, which don't overlap with the master, transformations can be concatenated, as long as they can be calibrated relative to some other camera. Evidently, this might significantly reduce accuracy of the calibration, however, for many applications a good calibration relative to the next sensor will usually be sufficient. Good calibration of the sensors requires some effort, because multiple pictures have to be taken with the calibration pattern in different positions and these have to be manually annotated or deselected, where automatic detection of the pattern fails. Calibration of one camera relative to another can therefore take up to an hour. Alternatively, other methods of calibration incorporating visual features, the depth map or prior knowledge of the environment could be used to improve the global registration of the cameras. Kainz et al. [64] for example suggest a method combining tracking values of 2D markers and depth measurements to compensate for variations in depth values of individual sensors. Results of [64] indicate that for accurate surface reconstruction the refined calibration strongly improves the results, because deviations of few centimeters already result in distorted 3D models. However, for our use case improvement might be less significant, because of the error introduced by the skeleton model fitting process [97], which in many cases can be higher than variation of depth between sensors.

5.4.2 Design

Our flexible combination of MoCap data enables us to almost arbitrarily scale the tracking volume. To achieve this, we are using a local area network to connect different sensors and to distribute tracking data. Furthermore, sensors are grouped in a modular way and arranged in a tree-structure connected by the network. Possible arrangements are shown in Figure 5.2. The different elements in the structure of our system are called cell and node. A cell consists of a single sensor together with associated software components to derive human pose data from it. A system diagram of a cell can be seen in Figure 5.3 a). It is the basic structure of our system.

A node is placed on top of one or more cells, collects tracking data and manages interaction between its cells. It communicates with other nodes and is therefore also responsible for network communication. A diagram of a node is depicted in Figure 5.3 b). One of the most important tasks of a node is the merging of MoCap data from different cells. It waits for input from all cells, which are currently tracking a user, and then utilizes the collected data to create the most probable skeleton pose. How different skeleton poses are fused is described in the next section. Usually, for reasons of applicability (e.g. length of sensor cables), a node will also be associated with the spatial structure of the building (i.e. a node will usually not extend beyond a room).

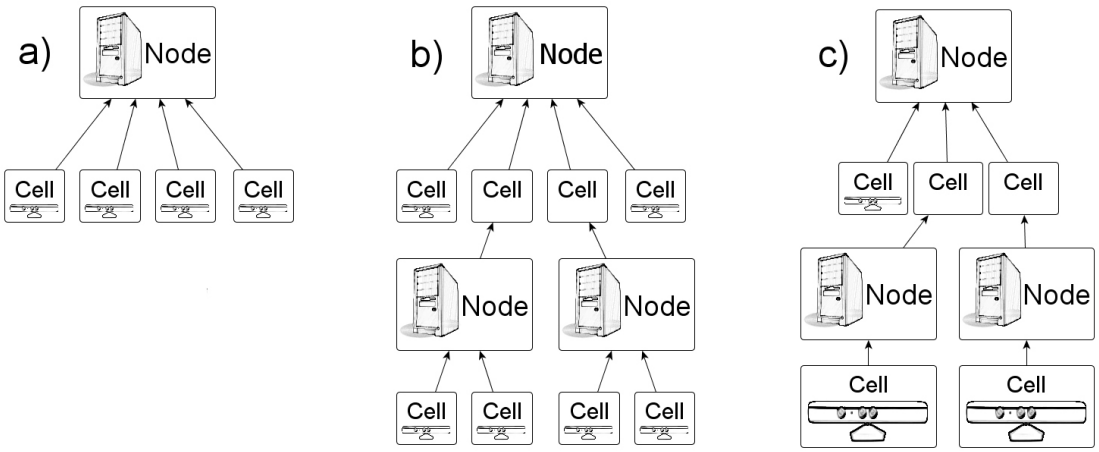


Figure 5.2: Arrangement of cells and nodes. a) node with local cells b) nodes with local and networked cells c) nodes with one cell each connected to a root node via network

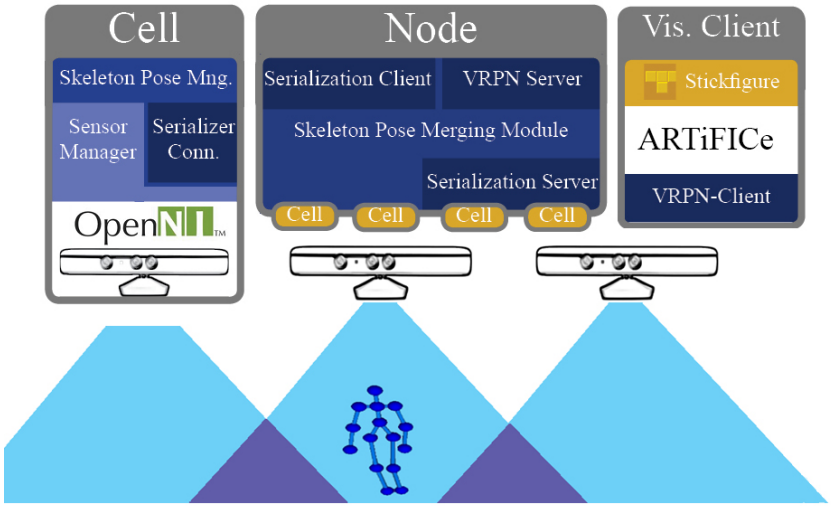


Figure 5.3: System diagrams of the different elements of our system. a) cell b) node c) visualization client.

The root node of the tree-structure has a special task. It collects and merges data from all nodes and makes it available to other applications. The elements described above enable us to create flexible network structures as depicted in Figure 5.2. The elementary setup consists of one node only with one or more local cells connected to it and is depicted in Figure 5.2 a). In a more elaborate configuration, multiple nodes are connected to cover a larger area as shown in Figure 5.2 b). With this setup multiple rooms can also be covered easily. Finally, Figure 5.2 c) shows a special case, where each node has only one local cell attached. This is the configuration we used in our tests for reasons described in the next subsection.

5.4.3 Implementation

Our system is implemented in C++ and uses OpenNI version 1.1.0.39 [110] to retrieve data from the sensors. NiTE (version 1.3.1.4) [109] provides software implementation for all OpenNI modules including the user generator. This module performs human skeleton tracking on a per-cell basis. Furthermore, we make extensive use of the Boost [16] library for various purposes like serialization and network communication.

5.4.3.1 Cell: Acquisition of Pose Data

Each cell encapsulates OpenNI modules, which are necessary to retrieve user and depth data. The latter is used purely for visualization. We implemented custom data structures for the use and serialization of data. Data is distributed over the local area network using TCP-connections. Two types of data have to be distributed: calibration data and skeleton pose data.

Initially, the skeleton of each user must be calibrated to reflect his real measurements. This has to be done only once while the user is within the tracking area. To perform the calibration the user has to stand in front, facing the sensor in Y-pose for about half-a-second as described in [110]. Once finished, the skeleton calibration is handed over to the containing node for distribution to other cells. This accounts for the version of the NiTE-middleware used in the implementation. Later versions have been released where no calibration pose is required. Skeleton pose data is generated in all cells for every frame, whenever the sensor delivers an update. This is also the case for frames, where no user is detected by a sensor and an empty pose is generated. Updates in every frame are important for the merging process as discussed in Paragraph 5.4.3.2 b). The cell sends the pose to the node, in which it is contained, for further processing. A schema of a skeleton pose object is depicted in Figure 5.4. It contains translation and rotation for each joint and a confidence value for each transformation, as delivered by OpenNI. In addition, for each joint transformation of the pose we add a weight attribute indicating the number of sensors used to generate it. This is important for the merging process, where a pose calculated from two sensors should have more importance than one from just one sensor. For one user the payload data of a pose update can be up to approximately one kilobyte. Therefore, the bandwidth required for the transmission of this data is rather small. Furthermore, in most cases the pose will be empty for a larger environment, since a user will only be tracked by a small number of cells.

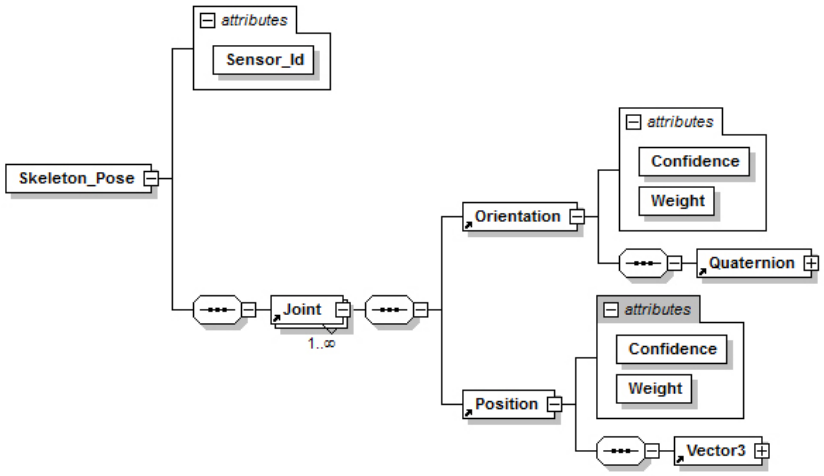


Figure 5.4: Schema of our skeleton pose structure

5.4.3.2 The Node: Merging and Communication

As stated before, the node has two major purposes - merging of MoCap data and communication.

a) Communication For communication, each node has a client and server available, which can establish TCP-connections between nodes. The client can be used to connect to one node higher up in the hierarchy (in the direction of the root). The server, on the other hand, can establish multiple connections with other nodes one level below in the hierarchy. One connection is established per node. It is communicated to the rest of the nodes as a networked cell, with the same properties as a local cell, only that it streams data from the network instead of from a sensor. Configuration of the connections of each node is established via a local XML-configuration file or application arguments. Once the nodes are connected, the network can be used to serialize and pass objects between them. The skeleton calibration is distributed from the root node via optional intermediate nodes to all cells. Therefore, it has to be performed only in one cell of the root node. The calibration is then automatically stored in a file and distributed to all cells. While local cells are loading the calibration from file, for distributed cells the data is serialized over a TCP-connection.

MoCap data is collected in a node in the form of the pose objects described in Section 5.4.3.1. No hardware synchronization of multiple sensors is available. For local cells synchronization of the cell data can be easily achieved on the software side by the mechanisms described in [110]. For cells receiving data over network connections, the transformations are updated as soon as they arrive and processed in the next loop of the merging process. If no update arrives for two frames, the old data is considered invalid. If an empty pose object arrives, the data of the corresponding sensor is also invalidated. Empty pose objects are otherwise ignored in the merging process. Due to the delay in the network communication the latency is slightly increased with networked cells. However, the small packet size of the pose updates limits network load and ensures fast updates.

b) Merging Once a node has received updates from all connected cells (local and networked), it attempts to merge the MoCap data into one skeletal pose. This fusion is performed on a per-joint basis taking into account the confidence and weight attributes. Position and rotation of a joint are treated similarly, except that position vectors are linearly interpolated and for the rotation quaternions spherical linear interpolation is used. Therefore, we will discuss only position merging in the following.

First, the positions are sorted according to confidence. If only one position with the highest confidence is available, it is returned. Otherwise, starting with an empty combined position, the positions from the cells are added one by one. This is done by linear interpolation between combined and new position. The weight-ratio to the combined positions determines the contribution of the new position in the interpolation. After each added position, the weights are updated for the combined position. For positions this can be considered as the weighted average position.

However, for future work we want to keep the implementation more flexible so more elaborate merging strategies can easily be incorporated, which might also take into account positions with less confidence or other attributes. Nevertheless, in the current NiTE implementation confidence values are limited to 0, 0.5 and 1 to distinguish between low and high confidence. Only transformations with a confidence of 1 were useful for our purpose, because lower confidence usually is associated with a very rough approximation. This largely decreases the options for merging the data. Once position and orientation are merged, they are updated for each joint in the merged skeleton pose, which is then sent up the hierarchy to the root node. Once all updates have arrived and are merged at the root level, tracking for the whole tracking area is complete and can be passed to the application.

5.4.3.3 Interfacing Applications

Nodes also contain an interface, which can be accessed by applications. This interface is usually only active for the root node and can be configured via XML-file. It consists of a VRPN-server [147], which sends updates for each frame and has a station defined for each joint transformation. Applications based on our ARTiFICe framework, as described in chapter 6, are using the VRPN-client implementation of the ARTiFICe middleware layer to access and utilize data. For this work we have implemented a small application based on ARTiFICe, which applies the transformation to a stick figure for visualization purposes. A schematic overview of this visualization client can be seen in Figure 5.2 c). We utilize the functionality of the ARTiFICe core classes, which receive joint pose and transformation updates for the joints and can map them to game objects or avatars.

5.4.4 Results

We have evaluated our system with different sensor placements to simulate situations which occur in wide area tracking. The test setup consists of three nodes with one cell/sensor attached to each, as depicted in Figure 5.2 c). The nodes are run on three different Windows-PCs with moderate hardware – two dual-core notebooks and one quad-core PC. Currently, only one sensor can be used per PC in our setup. This is caused by the fact, that each sensor needs a separate USB-host controller but mainly due to limitations of the current NiTE implementation. Its user

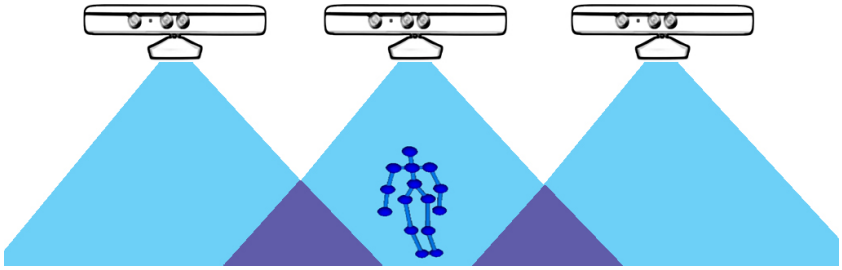


Figure 5.5: Sketch of the setup with three sensors

generator only works properly with one sensor per PC. The quality of the merged tracking data is, of course, to a large degree dependent on the accuracy of the posture delivered by NiTE to each cell. A short evaluation of the accuracy of the tracking data produced by one Kinect can be found in Section 5.3 and has been published in [131].

Furthermore, the quality of fused data depends on the extrinsic calibration of the sensors. With the method and tool presented in Section 5.4.1 we have achieved calibration accuracy of a few centimeters. This is due to the sensor's limited shared field of view. However, for most applications in entertainment, e.g. animation of an avatar, the current calibration accuracy is sufficient, since with the sensor's given limited tracking accuracy it doesn't produce new visible artifacts.

5.4.4.1 Completing Tracking Data in Overlapping Regions

Here, we are taking a look at an overlapping area of two sensors. At the edge of the view frustum it can occur, that single limbs are already outside the field of view of the depth camera and are therefore not properly tracked. However, as long as the torso is visible, a good estimate for the rest of the body is usually possible. In case the missing limb is visible in another sensor, the pose can be easily completed. Therefore, placement of the sensors is crucial. The overlap has to be large enough, so the seamless transition from one cell to the other is possible. On the other hand overlapping sensors produce disturbance (due to overlapping projected dot patterns) and therefore should be kept to a minimum. Figure 5.5 depicts the setup for this test of sensors placed side by side with a small overlap.

Figure 5.6 shows, how each arm is only visible in one sensor. Also, one leg is only tracked by one node. After merging the data our visualization client uses the completed correct pose for animation of the stick figure.

5.4.4.2 Extending the Tracking Area

1) Increasing the volume To increase the tracking volume three sensors are placed side-by-side in a larger area with small overlapping regions. This setup could also be arbitrarily extended. Figure 5.5 shows the setup of the three sensors, where sensors have been placed at an approximate height of 1.3 meters and 2.5 meters apart. Experiments with fixating them in higher positions and tilting downwards strongly decreased tracking quality of the lower extremities. To our experience level alignment works best. Overlap was about half-a-meter at the range of the

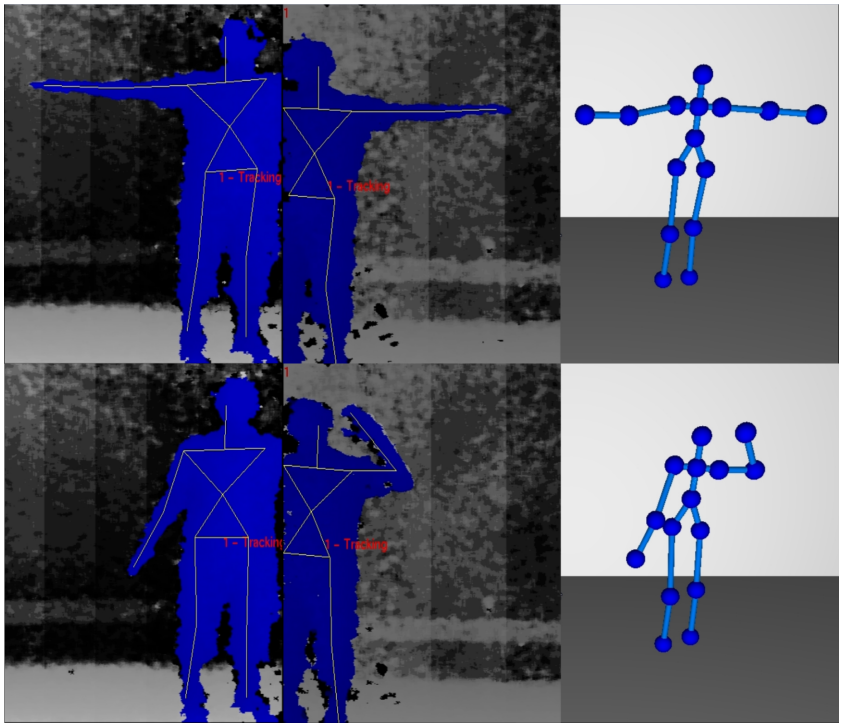


Figure 5.6: Demonstration of mutual completion of the pose by merging data from different nodes. Visualizations show the tracked user in the depth images of two sensors and corresponding skeleton pose as mapped to the stick figure by our visualization client.

tracked user. Figure 5.7 depicts snapshots from the depth images and avatar animation, while a user is changing from one cell to another. Note that the animated avatar in the bottom part of the figures appears slightly tilted in the areas on the side due to perspective projection. Facing the sensors and stepping sideways from one sensor's field of view to the next works very well, because in this position there is a lot of data available to infer the user's posture. Tracking a user from the side, on the other hand, usually produces not the best results, because one arm and leg are at least partly occluded by the rest of the body. Therefore, limited posture information or poor confidence is often produced. In this scenario it can hardly be compensated for by merging data, because all sensors have similar perspectives on the user. Placing sensors orthogonally, as described in the following subsection, greatly improves tracking quality in these cases.

2) Extending from room to room For the final tests we were taking a look at the possibility of covering a larger indoor environment with multiple rooms. We have therefore placed two sensors inside the room and one outside the door orthogonally to the other two sensors as depicted in Figure 5.8. Figure 5.9 shows the correct transition from one room to the other. Placing sensors normal to each other also improves tracking of movements, which can only be captured poorly from some perspectives e.g. tracking walking from the side is difficult due to occlusions. However, small pathways make an exact extrinsic calibration of the sensors difficult and sometimes

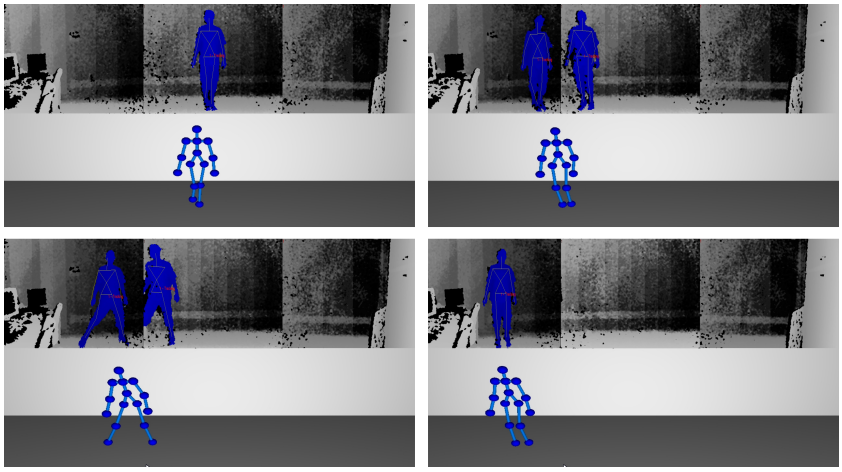


Figure 5.7: Different snapshots of one user changing from one cell to another.

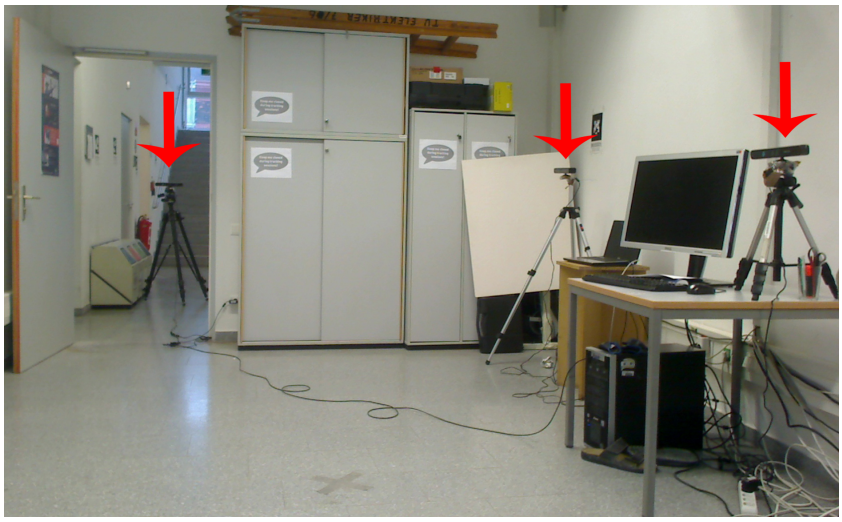


Figure 5.8: Photo of our setup with one sensor in the next room.

result in artifacts due to limited overlaps.

5.5 Discussion

We have evaluated our low-cost wide area MoCap system in different situations, which frequently occur in multi-camera setups. Merging the skeleton pose at a higher level (as opposed to merging the depth data before a skeleton is fitted) improves tracking data in many cases. In addition, it makes the system easier scalable. A video[1] shows the system in action.

[1] https://www.youtube.com/watch?v=qWSay6Cc840

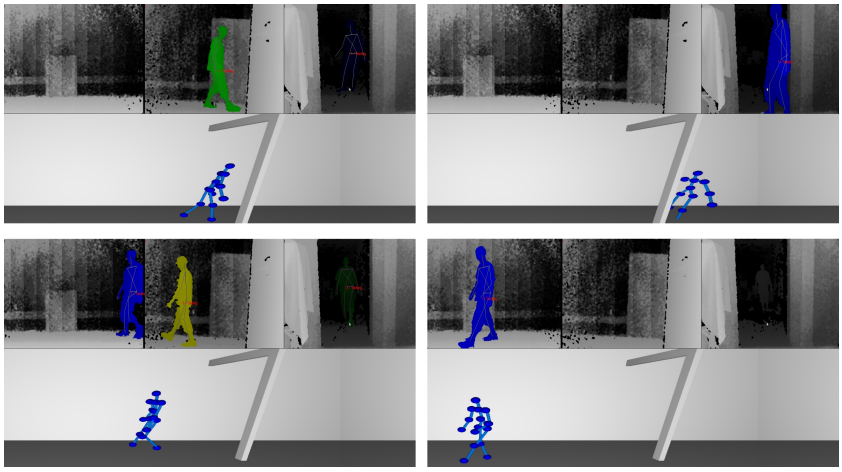


Figure 5.9: Snapshots of a tracked user from setup with orthogonal sensor placement in two rooms. Left side shows how multiple perspectives improve tracking. Right side shows the user in two different rooms.

For extending our work to the tracking of multiple users we would have to keep track of the global position of each user and identify each newly detected user in a cell by a global id. This plan is somewhat hindered by the fact that the skeleton calibration produced by OpenNI is only available in a binary file of one megabyte or above. Therefore, serialization and transmission takes a considerable amount of time in our current implementation. While a waiting period of 20 seconds might be acceptable for a single user, longer breaks with every newly added user in a multiuser environment would be very disruptive. Later versions of OpenNI/NiTE and Microsoft's KinectSDK allowed new users to be tracked without any specific calibration pose, a feature that would significantly improve usability especially in a multiuser system. The KinectSDK has no estimate for the confidence of the tracked joints, although recent versions of the KinectSDK differentiate between joint data that is not tracked but "inferred" by calculation from other tracked joints and actually "tracked" joint data. Combining data from multiple sensors using elaborate pose merging strategies incorporating movement trends and stronger emphasizing confidence values would be promising. However, with the acquisition of Primesense through Apple and associated discontinuation of support for the NiTE middleware, there seems to be no immediate opportunity.

CHAPTER 6

A Framework for VR/AR and Multimodal Feedback

Developing applications and serious games for rehabilitation, especially with a VR setup, requires a lot of time and effort, because in addition to the implementation of game logic and content, often input/output devices and technologies have to be integrated and their data processed. Support for integration of non-desktop devices is usually not available in standard game engines. With a suitable framework on the other hand the development process can be significantly simplified and sped up. Therefore, for serious games in rehabilitation and other applications in research and teaching we have developed a Virtual and Augmented Reality (VR/AR) framework - ARTiFICe [93] -, that is lightweight and flexible but still powerful. Furthermore, it is extensible to easily integrate new devices and technologies. In addition the framework provides software modules that eases development of 3D interaction techniques and distribution in the application. These features together with the integration of an off-the-shelf game engine makes it very well suited for the development of serious games.

The ARTiFICe framework also facilitates development of collaborative AR applications and interaction techniques for mobile devices employing Vuforia [111] for tracking. However, hand held AR scenarios are not the focus of this thesis. The interested reader is kindly referred to [93] and [92] for further details. In contrast to the latter, this chapter focuses on integration of MoCap data, biosignals and haptic feedback.

This chapter is organized as follows. Section 6.1 gives a short motivation followed by related work in Section 6.2. Section 6.3 introduces the components of a VR/AR system in general. An overview of the framework architecture and data flow is provided in section 6.4 and detailed in sections 6.5.1 through 6.5.2. Section 6.6 describes the integrated devices and how they interface with the framework, while Section 6.8 describes interfaces to external tools. Section 6.7 provides information on the workflow of creating a new application, while Section 6.9 gives an overview of setups and application developed based on the framework.

6.1 Motivation

For the development of serious games in rehabilitation and other applications in VR/AR a framework should be flexible, extensible and should provide good usability at low cost. Existing toolkits and approaches, as described in the next section, have various drawbacks regarding these dimensions. Therefore, we decided to develop a framework based on an off the shelf game engine for collaborative and distributed VR/AR applications supporting multiple users and various input devices for interaction.

Overall our framework provides the following features:

- An extensible middleware layer providing abstraction of input devices and convenient interfaces (Section 6.5.1).
- An application layer implementing reusable and adaptable modules for 3D event handling, interaction and distribution that can be used in collaborative applications for different VR/AR setups (Section 6.5.2).
- Integration of various input devices such as 2D fiducial markers, 3D mice, video game controllers, depth cameras, 6 DOF targets, biosignal acquisition systems (Section 6.6 in Subsections 6.6.1-6.6.4).
- Support of a range of output devices e.g. stereo monitors, stereo projectors, head mounted displays (HMD), smartphones and haptic feedback devices (Section 6.6 in Subsections 6.6.5-6.6.6).
- A graphical user interface and scene management for authoring and rapid prototyping of VR/AR applications (Section 6.7).
- A number of flexible interfaces to external applications (Section 6.8)
- Support for versatile VR/AR setups (non-immersive, semi-immersive, immersive) on different operating systems and platforms (Section 6.9).

6.2 Related Work

Since the mid-1990s, a large number of VR/AR frameworks have been developed and a variety of systems supporting distributed VR applications emerged [55]. Most software frameworks are based on scene graph libraries, for example open source toolkits such as Studierstube [123], VR Juggler [31], Avango [76] or commercial ones like 3DVIA Virtools [154]. They provide varying support of multiple input and output devices. In the following, we briefly discuss frameworks related to our work.

Studierstube is an application framework for collaborative Augmented Reality. Its development started in 1996 and continued for almost ten years while it was used for research and teaching. It simultaneously supports multiple users as well as multiple applications, which are embedded as nodes in a scene graph. While this open source C++ based framework is very powerful, it is hard to maintain, does not provide a graphical user interface for scene management

and is difficult to learn within a short period of time, which is important if used for teaching. In addition, recent technologies and devices such as mobile phones or depth imaging sensors are difficult to integrate.

3DVIA Virtools is a commercial development and deployment platform for interactive 3D content creation. It supports multiple users and physics behaviour to create immersive and distributed applications using industry standard VR peripherals. It offers a comprehensive graphical development environment and can deploy to a wide range of output devices. However, its application for research and teaching is limited due to high licensing costs.

One of the first AR frameworks using off the shelf software to design and develop AR applications was DART [86]. DART is based on the Macromedia Director multimedia programming environment. It uses the familiar Director paradigms of a score, sprites and behaviors to allow a user to visually create complex AR applications. DART also provides low-level support for the management of trackers, sensors, and cameras via a Director plug-in Xtra. However, DART is not suitable for research and teaching due to licensing costs for Director. In addition, it lacks stereo output support and the time line based scene management is rather made for story telling environments than VR/AR applications. Our ARTiFICe framework on the other hand supports a number of different VR setups as described in Section 6.7.

Similar to Virtools, Unity3D [151] features an editor for authoring 2D and 3D content and comprises a game engine for executing the application. Current license models make Unity attractive for personal and academic use. Nevertheless, Unity3D by itself is no VR/AR framework but is designed for creating 3D video games and other interactive content. It offers a powerful rendering engine providing lighting, physics, network communication for collaboration and content distribution. Furthermore, it provides an integrated programming environment using C#, JavaScript or Boo while development can be done under Windows as well as Mac OS X. The final application can be built – generally without changes – for various platforms such as Windows, Mac, iOS, Android, all major game consoles, Flash and web clients. Although some sophisticated plugins exist for Unity3D integrating e.g. VRPN devices [159], a unified approach for device integration and many other features of a VR/AR framework are missing so far. Our framework in large parts uses Unity3D as its underlying development platform and rendering engine. Most VR/AR specific extensions were built around Unity3D as described in Section 6.5.2.

6.3 Components of a VR/AR system

The five classic components of a VR system according to Burdea et al. [22] are the User, Task, I/O Devices, VR Engine and Software&Databases. This still holds true for most modern setups, but emphasis has shifted due to new developments. While [22] states that "special-purpose computer architecture designed to match the high I/O and computation demands of real-time VR simulations", technological advances during the recent years have made it possible to run VR applications on standard PC (and even mobile) hardware. Larger flexibility of hardware together with the emergence of the VR/AR frameworks described in section 6.2 has led away from the concept of building monolithic VR-systems for a single purpose. Figure 6.1 takes into account these recent developments and shows the architecture of a modern VR/AR Framework decou-

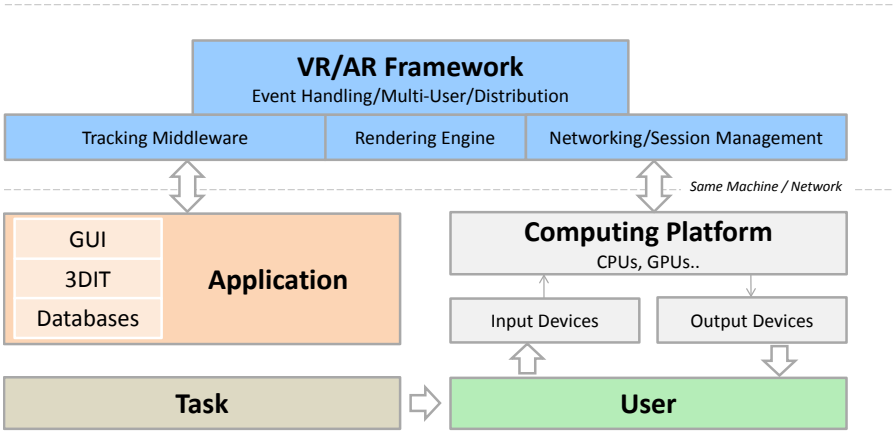


Figure 6.1: VR/AR Reality System Architecture.

pled from the Computing Platform rather than the VR Engine combining both as suggested by Burdea. The central part is of course the user (or multiple users) following a certain task, while interacting with the virtual environment. Therefore, human factors play an important role in the design of any VR/AR system and framework. Stanney et al. identify three primary subtopics: human performance efficiency in VR; health and safety issues; and potential social implications of VE [141]. In the context of our VR/AR framework human performance efficiency is the most important topic, since it incorporates a number of factors, which are influenced by the framework. Thus human sensory physiology has to be taken in account e.g. visual perception for designing visual presentations. Also sensory integration issues in multimodal interaction can be accounted for in the framework when integrating visual, haptic and audio devices. Finally, well-designed metaphors and 3D interaction techniques can be intuitive to a novice user and help in effectively carrying out a task [141]. In addition, the task itself might influence efficiency, with different tasks being more or less suited to be performed in VR/AR.

The system hardware (shown in gray in Figure 6.1) comprise input and output devices whose spatial position and orientation might be tracked and a computing platform with fast interfacing and powerful graphics processing capabilities.

As stated earlier developing serious games for rehabilitation or other VR/AR software applications can be tedious work without proper authoring support by the underlying framework. Many modeling tasks have to be performed (geometric, kinematic, physical), I/O data have to be mapped, some form of scene management is required and intelligent behavior (game logic) has to be implemented [22]. Furthermore, data has to be retrieved from or written to databases in many cases. Support for these tasks should be provided for in the framework through an editor

for authoring 2D/3D content and reusable components for 3D interaction and data access.

The VR/AR software framework handles I/O data, and optionally distribution of data over network (for multiple users or for remote I/O devices). Various input devices and especially technologies for tracking are in use to determine the location of input and output devices as well as specific body parts of the user up to full body motion capture. Input data of these devices is received by the computing platform (e.g. workstation, mobile device) and handed over to the VR/AR framework. Convenient interfaces are usually provided by a (tracking) middleware layer. The middleware processes and transforms input data to provide it for subsequent usage within the application. Based on this input data the user can employ provided 3D interaction techniques. Recent low-cost video game controllers as well as powerful mobile hardware offer great opportunities by supporting multiple users in interactive collaborative virtual and augmented reality applications. If multiple users work together, communication is controlled by a network layer while 3D interaction is handled by an event handling mechanism. Subsequently, the virtual scene is visualized to the user on its output device using the rendering engine.

Ideally, a VR/AR framework should offer high quality real-time rendering, physics support, networking and scene management to build rich 3D applications. Additionally, for research and teaching purposes, a virtual reality framework must be inexpensive, quick to familiarize with, well documented and flexible for feature extension and rapid integration of novel hardware solutions.

6.4 ARTiFICe Architecture Overview and Data Flow

Our ARTiFICe framework follows the basic structure of a modern VR/AR framework introduced in the previous section. In Figure 6.2, an overview of the framework with its components and the data flow is illustrated.

The framework has been developed as a loosely coupled modular system, thus it is largely independent of the computing platform and can be easily extended to support novel I/O devices. Input data from various devices is fed into the framework using a transparent and adaptive middleware layer (Section 6.5.1). The middleware transforms all input data in a consistent way and delivers it to the application layer through comprehensive interfaces.

The application layer (Section 6.5.2) is built on top of an external game engine. Within the application layer, the ARTiFICe core handles the 3D event data, offers distribution support and delivers the data to the game engine's scene management. Furthermore, human performance efficiency, as introduced in the previous section, is enhanced by providing interaction techniques as adaptable and extensible modules. To provide various 2D as well as 3D interaction techniques for use in applications, the ARTiFICe core offers multiple implementations of interaction metaphors and is easy to extend with novel concepts (Section 6.5.2.5). The interaction module offers a single interface to process 3D event data from an input device to control the virtual interaction device and to manipulate virtual content. In addition, multiple haptic feedback devices are integrated in the framework. They can be controlled through different easily accessible haptic output modes available through the application layer (Section 6.5.2.7), that have been designed to follow human sensory physiology and skin perception. The virtual scene with real-time inter-

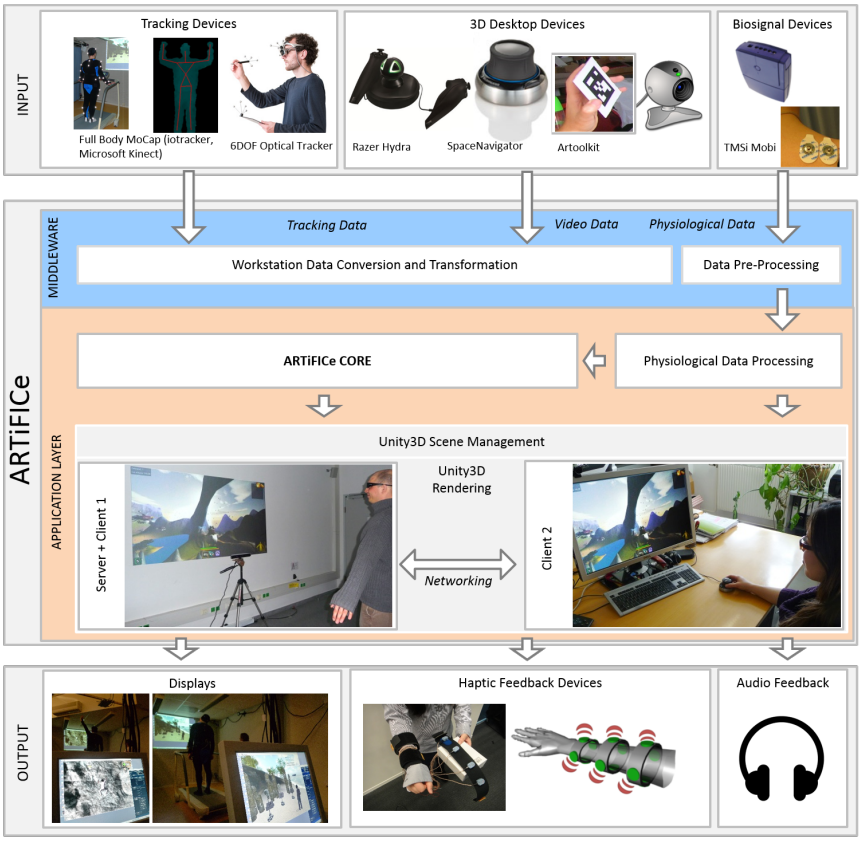


Figure 6.2: ARTiFICe framework components and data flow.

action is then visualized on different output devices using the game engine's rendering module. Furthermore, 3D audio output is generated by the game engine.

Besides interaction with 3D content, the co-presence of multiple users interacting with the same content at the same point in time opens up great possibilities for collaborative tasks. Hence, we integrated a distribution framework into the ARTiFICe core to enable real-time user-managed collaboration for various hardware setups for two or more users over the network (Section 6.5.2.6).

Unity3D offers a powerful rendering engine and the possibility to extend its functionality through scripts and plugins to access devices described in Section 6.6. In addition, the powerful editor of Unity3D makes it well suited for the various modeling tasks introduced in the previous section and extended in Section 6.7. Furthermore, Unity3D provides a free to use license that includes all features we need for scene management, authoring, rendering and physics support. Further details about the ARTiFICe core are given in the next sections.

The following section describes the design and implementation of different components of the framework in more detail.

6.5 ARTiFICe Framework: Design and Implementation

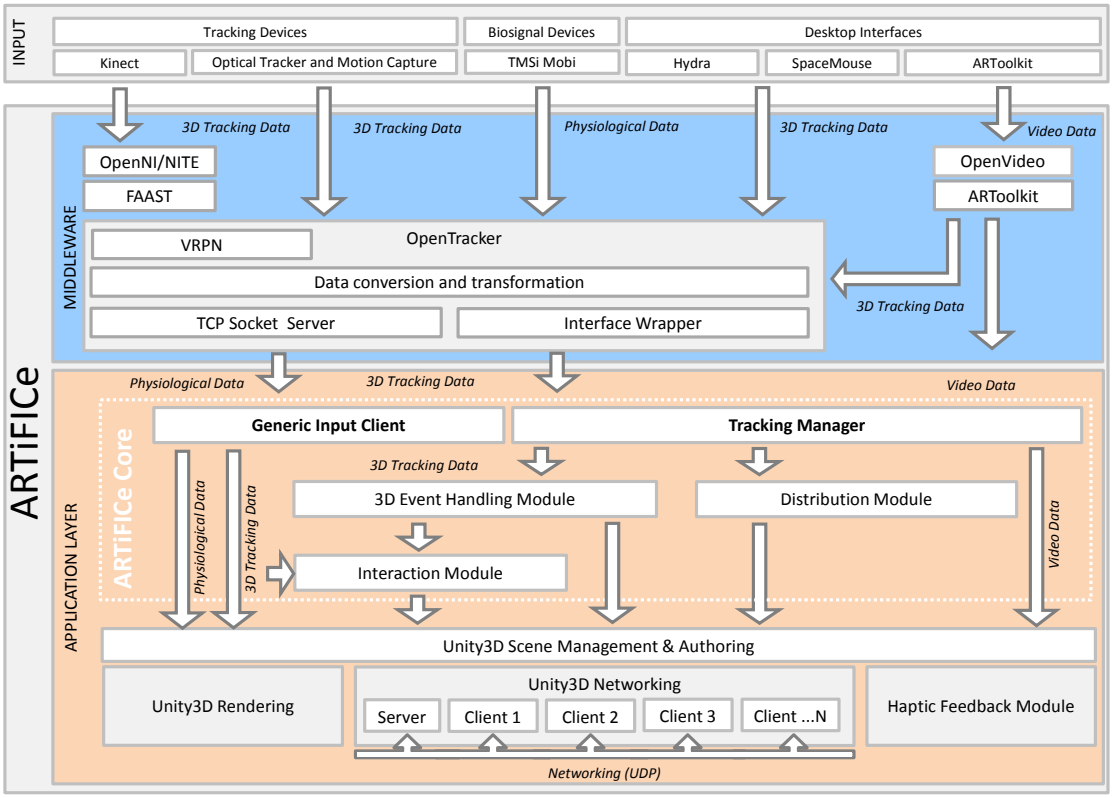


Figure 6.3: Detailed framework components.

This section presents details on the design and implementation of our ARTiFICe framework, which follows the architecture of a modern VR/AR framework as introduced in the previous two sections. Figure 6.3 shows a detailed view on our framework with its data flow and components. The main elements of the framework are the middleware and the application layer highlighted in blue and orange respectively. The following subsections provide details on these two layers (middleware layer Section 6.5.1 and application layer Section 6.5.2).

6.5.1 Middleware Layer

6.5.1.1 Design

The middleware in our ARTiFICe framework has to serve multiple purposes. Firstly, as can be seen in Figure 6.3, it integrates many different input devices. Secondly, it pre-processes input events of these devices including 3D event data, physiological data or video data. Thirdly, data has to be passed to the application layer through consistent and convenient interfaces, so applications are independent of changes in the hardware setup.

6.5.1.2 Implementation

To gather and process the input of various devices in one single software layer, we use OpenTracker (OT) [114], as introduced in Section 2.8. It is an open source (tracking) middleware, which offers transparent and flexible device integration, pre-processes input events and passes them through a multi-modal data flow network to the application layer. It is framework providing functionalities required for tracking in VR/AR applications and easing the development and maintenance of hardware setups. Furthermore, it can transform and extract data (e.g. information about velocities, specific positions etc.) from the skeleton pose data produced by the tracking device.

The class central to OpenTracker is *Context*, which runs the main loop, configures and manages all modules and the data flow as described in Section 2.8. In the ARTiFICe framework an OT *Context* can be either run in

1. a separate process or in

2. a separate thread of the application.

Method 1 uses network communication (TCP sockets) to exchange data between OpenTracker and the application layer using a custom XML protocol described in Section 6.5.2. In both cases a new OT node called *UnitySink* provides a single sink for all input devices, which caches data for transfer to the application layer. For the first case *UnitySink* encapsulates a TCP socket server and a serializer, which generates XML strings from the input events. The server thread then sends the XML-packets to the client on the application side.

For the second case synchronized methods are available for the application-thread to pull data from the memory shared with *UnitySink* directly. The *UnitySink*-node is then referenced directly during run-time by the ARTiFICe core, which provides the fetched data to the application.

Both interfaces have their merits and drawbacks, which makes them more or less suitable in certain contexts. Method 1 provides better decoupling between the layers, which has been a design goal for our framework. Especially for testing of the complex setup described in Chapter 7 this interface has shown to be beneficial because input of devices could be easily simulated by generating appropriate XML packets or redirected to a different test application. Furthermore, interface 1 is more flexible in terms of data and devices as can be seen in Section 6.5.2.4. It can support generic input as long as the input device and data format has been implemented on the application side.

Multiple devices have been implemented as described in Section 6.6 supporting not only tracking, but also MoCap, biosignal acquisition systems, speech input and even results from emotion recognition. Data formats required for some of these input devices are not natively supported by OpenTracker and could therefore not be distributed over network. However, especially for input which is typically generated on dedicated machines (e.g. MoCap), this can be crucial. Nevertheless, data serialization and parsing adds an additional overhead to the processing queue and might therefore not be desirable, especially on devices with less computational power (e.g. mobile). Also for less experienced users/developers or an application area with less demands on

flexibility regarding device integration (e.g. desktop AR/VR) launching the OT *Context* from the application is more straight-forward. Therefore, we have added Method 2 to the framework.

In any case, OT as middleware in ARTiFICe provides a transparent interface and loose coupling between the set of physical devices and the application layer, passing and processing tracking and physiological data from input devices.

6.5.2 Application Layer

6.5.2.1 ARTiFICe Core Design

The ARTiFICe application layer integrates multiple modules which provide core and extended functionality required in VR/AR applications, as illustrated in Figure 6.3. The core functionality provides interfaces to the application to access the middleware and modules for 3D event handling (Subsection 6.5.2.3), handling of generic input (Subsection 6.5.2.4), interaction (Subsection 6.5.2.5) and distribution (Subsection 6.5.2.6). In addition, a haptic feedback module (Subsection 6.5.2.7) allows the user to create and render haptic output parallel to the visual and audio output.

6.5.2.2 ARTiFICe Core Implementation

The ARTiFICe modules have been implemented as collections of C# classes, utilizing functionalities and based on the structures provided by Unity3D. All virtual objects in the scene are managed by Unity3D's scene authoring. Therefore, Unity3D provides a container class called *GameObject* to which geometry, transformation nodes, textures, physical properties as well as C# classes custom-implemented by the user can be attached to control visual appearance and overall behavior of a virtual scene object. Using a form of the Decorator Design Pattern *GameObjects* can be extended by so-called *Components*, that implement the behavior of an object. In addition to readily available *Components* the user can also implement his/her own scripts and attach them to *GameObjects*. We made extensive use of that concept in our framework. *GameObjects* can also be grouped in a logical manner forming a transformation hierarchy. For in depth explanations please refer to the Unity3D documentation [151].

Two software modules in the ARTiFICe core build the counter-pieces for transferring data from the middleware layer to the application layer, as described in Subsection 6.5.1.2:

1. The Generic Input Client
2. The Tracking Manager

The Generic Input Client implements a TCP socket client receiving data in XML format from the socket server contained in *UnitySink*.

The ARTiFICe Tracking Manager controls the 3D event dataflow between middleware and application layer. If OpenTracker is not run as a separate process, it reads the OpenTracker configuration files and loads the dependent tracking libraries at application start-up. It also starts an OpenTracker instance and stops OpenTracker at application shutdown. If ARToolkit+ is used, the Manager also parses the Openvideo configuration file and opens and closes an OpenVideo handler for marker tracking.

A detailed explanation of the implementation of the generic input client as well as the modules for 3D event handling, interaction, distribution and haptic feedback is given in the next subsections.

6.5.2.3 3D Event Handling Module

Design The 3D Event Handling Module receives 3D events containing a 6 DOF pose from the Tracking Manager and provides methods to the application to access these data in a convenient and efficient way.

Implementation The 3D Event Handling module feeds tracking data of an input device into the transformation *Component* of a *GameObject* to map a tracked real physical position and orientation to a virtual object. The overall concept of the 3D Event Handling module is shown in Figure 6.4.

The tracking base class inherits from the Unity3D base class *MonoBehaviour* to be able to attach the deriving classes to any virtual scene object. For the various tracking setups, a subclass for each supported device was implemented. The device subclass is attached to a virtual scene object. As soon as the application starts, *TrackProvider* creates ARTiFICe *Trackers* through the ARTiFICe *Manager*, which is implemented as singleton. Each ARTiFICe *Tracker* is interfaced to the corresponding OT *UnitySink*-node as described in Subsection 6.5.1.2 and thereby provides the input data of all used devices to the tracking framework. In addition, for 2D marker tracking we developed our own multi-marker tracking support to be able to track cuboid-formed 3D objects and determine its pose.

All concrete tracking subclasses provide a consistent tracking data layer, which can be used as tracker object for further processing within the interaction framework.

In addition, *Avatar* provides a convenient interface for using tracking data to animate a humanoid model through tracking data. The script manages a list of bones, which can be configured to receive tracking data e.g. from Kinect through VRPN. The user can flexibly map the tracking poses to *GameObjects*, which are part of an avatars skeleton hierarchy, thus animating the model.

6.5.2.4 Generic Input Client

Design A flexible interface for the input of arbitrary data generated by input devices or middleware is helpful for many applications, such as that described in Chapter 7. The goal of this interface is to provide the user with a maximum of flexibility, while keeping the required programming effort on the application side minimal.

Implementation In a client/server approach data is transfered from a TCP socket server in the middleware to a TCP socket client implemented in the application layer. In the application layer the data is parsed and distributed based on the content. Game objects within Unity can subscribe to information from the various input devices and can be customized to utilize the data according to the device. 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 objects in the game world.

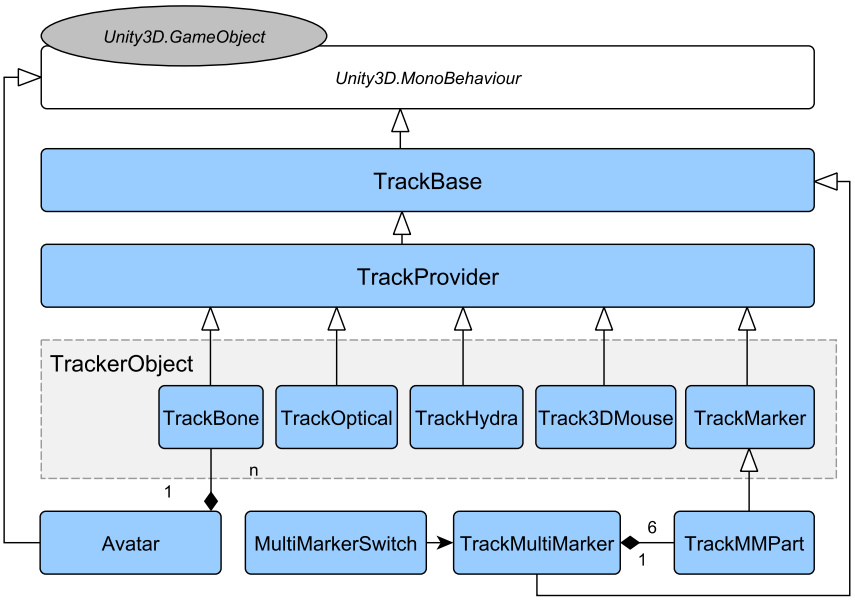


Figure 6.4: 3D Event Handling class hierarchy.

The data arrives in a custom XML format consisting of various XML elements. The root node states the type of data and child-nodes and attributes contain information like timestamps, confidence values, data primitives and combined types, vectors and others. Figure 6.5 shows two examples of schemas for XML elments for a) a generic single measurement value as used for e.g. EMG or heart rate and b) a rigid body pose with 6 DOF.

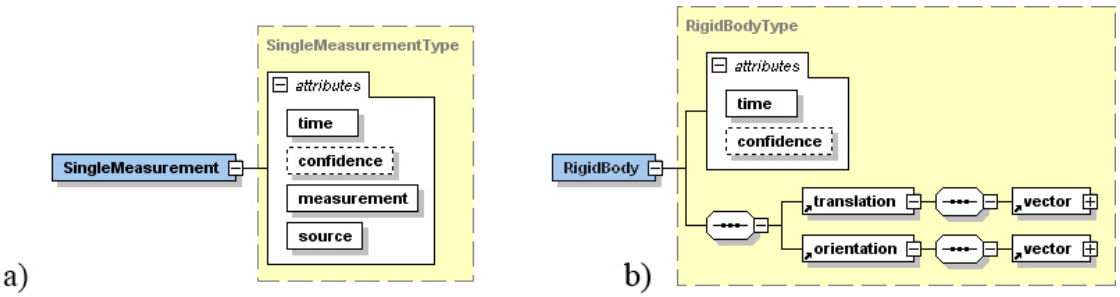


Figure 6.5: Visualization of XML Schema of the a) SingleMeasurement XML element and b) RigidBody XML element transferred between middleware and application layer.

The class hierarchy of these generic input devices is shown in Figure 6.6.

Each specific input device class inherits properties from a general *InputDevice*, which contains various functions for parsing XML packets and processing incoming data needed by all

of the classes. The specific device class then overwrites methods and adds variables and data structures that are device-specific. Each active input device is added to the list managed by *NetworkComponent*.

The class *NetworkComponent* is central to the architecture and handles the client side of the network communication on the one hand and manages a list of *InputDevices* on the other. *GameDataInterface* handles one or more Unity3D *GameObjects* with attached *GameControllers* and can be customized for every game or application. Depending on the data used, some devices have their specific implementation of *GameController* to map the data to a *GameObject* according to the applications requirements. Through *GameDataInterface*, *GameControllers* can subscribe to *InputDevices* in the list inherited from *NetworkComponent*. When new data is received, the monitoring *GameController* is notified and can read and use it within the application. How often this data should be monitored/handled can be adjusted according to the requirements of the game itself. The different *GameControllers* can also be connected to the *InfoController* and/or the *StorageController*. *InfoController* provides methods for storage and retrieval of user specific parameters (e.g. range of motion of the neck, which should be reached in a therapy application). *StorageController* on the other hand can be used to write log data and information required for later evaluation of game and input data, e.g. in a user study.

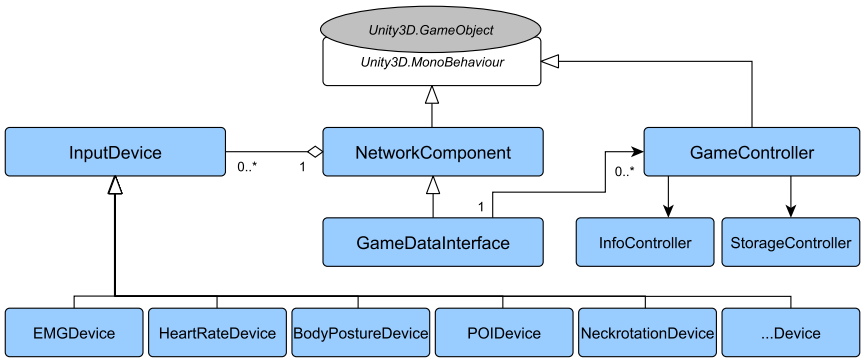


Figure 6.6: Generic Input Device class hierarchy.

Big parts of the functionality are being shared by the different 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 to distribute the computational load of data processing and analysis over different machines.

For the MoCap data the interface on the application side is described here in an exemplary fashion. *BodyPostureDevice* takes care of parsing the XML packets. During an initial phase the *NetworkComponent* receives structural information of the skeleton and once parsed by *BodyPostureDevice BodyPostureController* generates the joint composition of the body being monitored.

After this is complete, the rotation for a specific joint can be updated independently from the others. Furthermore, *BodyPostureController* creates and remembers the initial body posture of an avatar. 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 game world. Other *InputDevices* and Controllers work in a similar fashion, but usually without an initialization phase.

6.5.2.5 Interaction Module

Design A common task in VR/AR applications is that of 3D object selection and manipulation. Various Interaction Techniques (IT) have been successfully deployed in related work [108, 40, 17]. Usually, these techniques require tracking data input of a hand and head (or body) pose. The 3D Event Handling Module or the Generic Input Client provide the required tracking data while the development of specific techniques is simplified by templates and examples in the Interaction Module.

Implementation The basic architecture of our interaction module is illustrated in Figure 6.7. Raw tracking data is fed into the transformation *Component* of a virtual scene Unity3D *GameObject*, which subsequently can be used as input to interaction techniques (IT).

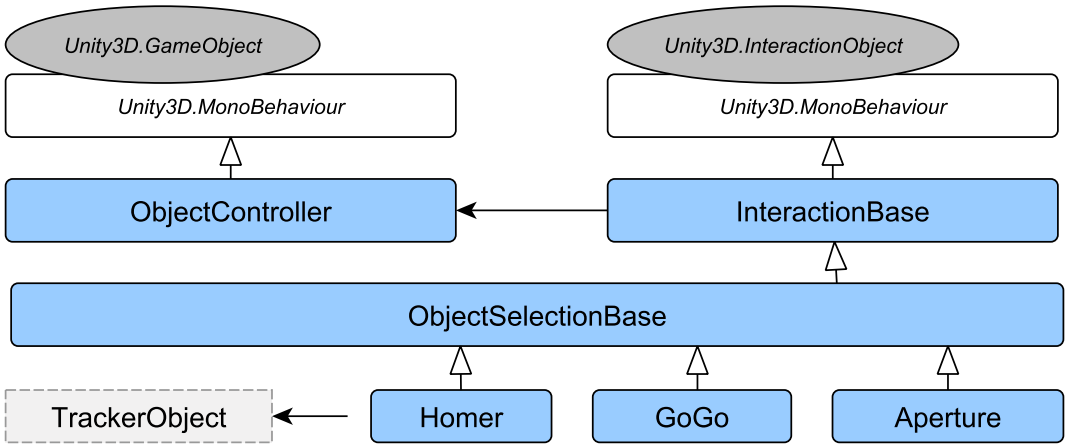


Figure 6.7: Interaction class hierarchy.

Using the 3D Event Handling Module as described in Section 6.5.2.3 we can map the raw tracking data to a specific *GameObject*, which will be referred to as tracker object. The interaction module then processes and uses the transformation of the tracker object according to the choosen specific IT. The abstraction layer *ObjectSelectionBase* provides a straight forward and clean interface of data handling and offers a transparent layer to integrate new techniques into the framework. The only information which must be handed over to *ObjectSelectionBase* is a list of the selected object(s) and the absolute pose of the interaction object, calculated by the

IT. This data is then processed by the *InteractionBase* class and delivered to all selected virtual scene objects. Virtual scene objects which should be selectable must have the *ObjectController* class attached. Depending on the given pose the *ObjectController* manipulates the position and orientation of the selected scene object.

As concrete 3D interaction techniques, our framework integrates several standard VR interaction metaphors, such as a simple VirtualHand, GoGo [108], Aperture [40] and HOMER [17].

6.5.2.6 Collaboration & Distribution

Design To provide multi-user support for interaction with different input devices and remote collaboration, we implemented a collaboration and distribution module. It is loosely coupled with the interaction module and enables distribution independent of the setup.

Implementation An overview of the distribution module and its connection to the interaction module is given in Figure 6.8. The *NetworkBase* class provides functions to initialize the server and to connect a client to the server. All connected clients are managed by the *UserManager* class, implemented as singleton. To reduce hardware necessary for realizing a client-server application and to improve overall usability, one device can act as server and client simultaneously.

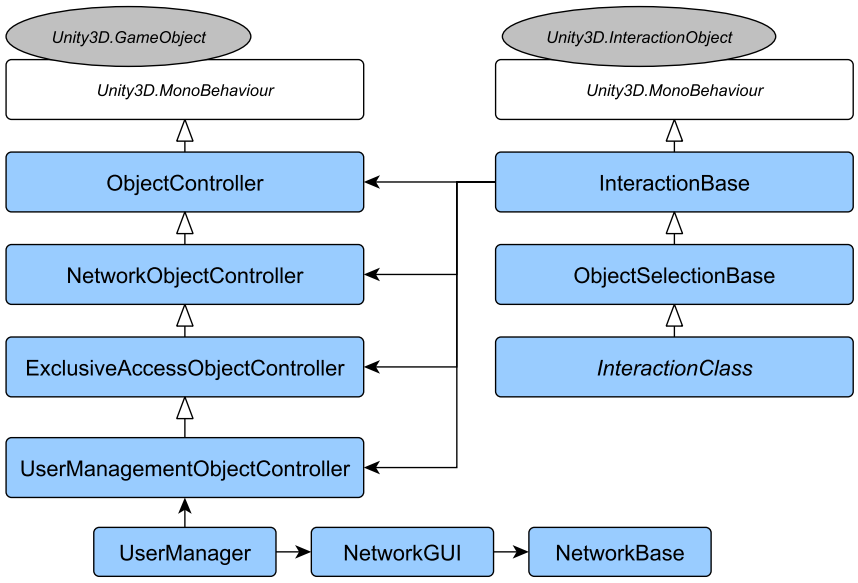


Figure 6.8: Distribution class hierarchy.

For distributed collaboration each selectable scene object must attach a *NetworkObjectController*, which distributes selection and manipulation functionality over the network. To enable exclusive access of a scene object *ExclusiveAccessObjectController* prevents simultaneous us-

age by multiple users. As long as a user manipulates the scene object it is locked for other users. To provide exclusive object access to a specific user the *UserManagmentObjectController* is used.

The network functions are based on the Unity3D network layer using UDP for communication. We are using a client-server architecture with a direct connection between the server and all clients (star topology). For data exchange remote procedure calls (RPC) and state synchronization are employed. To prevent data loss the state synchronization is buffered.

6.5.2.7 Haptic Feedback Module

Design Haptic output can aid to communicate the state of objects in a virtual scene or give feedback in motor learning or motion guidance tasks. Lindeman et al. [83] showed that providing tactile contact cues to users of VR systems can increase situational awareness. Furthermore, from visual feedback alone, interaction with virtual objects is often difficult, since depth is hard to judge even with stereoscopic displays, in cases where the sense of touch is missing. Furthermore, giving haptic feedback on pose, movement path, reaching or other tasks involving multiple DOFs can be applied in a rehabilitation scenario. Therefore, the haptic output module offers modes for three different functionalities:

1. Haptic augmentation of objects
2. Haptic position and rotation indication
3. Haptic indication of velocity

Haptic Augmentation of Objects

The haptic feedback module can augment interactions with virtual objects in an approach similar to [83]. We are using physics simulations to detect collisions with the user's body and generate 3 DOF directional tactile feedback. This module calculates direction and intensity of the feedback and activates the corresponding tactor, which can be used flexibly for different body parts and objects due to its low level of abstraction and complexity.

Haptic Position and Direction Indication

The rotation of a joint in the human skeleton has a maximum of three DOF and in a low-level, bottom-up approach, we could consider giving feedback on these three DOF to each limb in a similar and flexible manner.

The basic concept is to use two tactors to indicate rotation around one axis by activating a tactor on one side to push or pull the limb in a direction [82, 71].

Our haptic feedback module is used to guide a user's limb into a predefined, or interactively specified, posture using a minimum of two actuators per DOF, to indicate directional movement. This form of feedback can be used for example for driving arm movements or as navigational cues as illustrated in Figure 6.9.

An error function calculates the feedback strength and direction based on current and target rotation, and a second user (e.g. a therapist in a rehabilitation scenario) can adjust the feedback, or add/remove DOFs for different limbs.

Besides the primary vibrotactile feedback for this module, we also support pneumatic actuation, which provides a stronger tactile sensation. More complex mechanical configuration, however, limits its scalability to a larger number of DOFs.

Haptic Indication of Velocity

Interaction, e.g. for selection and manipulation as described in Section 6.5.2.5 is focused on either hands or arms in many applications, and it is therefore useful to encode more haptic information in these areas. This is challenging due to the limited area and resulting spatial and temporal interference.

Vibrotactors are, however, robust, relatively small and lightweight, consume little power, and can be used wirelessly and in spatial locality. Controllability of frequency and amplitude with quick actuation allows for implementations of different levels of abstraction and information encoding. This has made them popular for many applications, including our dense tactile display for an arm.

The sensory saltation effect can also be employed to add information like vectors or speed of intended movement. While the saltation effect has already been used to indicate rotations of the arm [82] or direction and rotation in a planar setting [61], indication of a vector in three dimensions on the arm, and especially speed, have been less explored.

Our module is designed to provide movement speed sensation in three directions by employing a dense tactile display, where speed is indicated by triggering the vibrotactors in sequence, as shown in Figure 6.10.

By controlling burst durations and onset times, perceived stroking movements can be generated at a desired target speed. The actuators are turned on for a pulse duration (t_{pulse}) of 20-200 ms, where 20 ms was chosen as the minimum speed which subjects could perceive as a moving tactile stroke in a pilot study. With a t_{pulse} of 200 ms, a single loop of indication would take

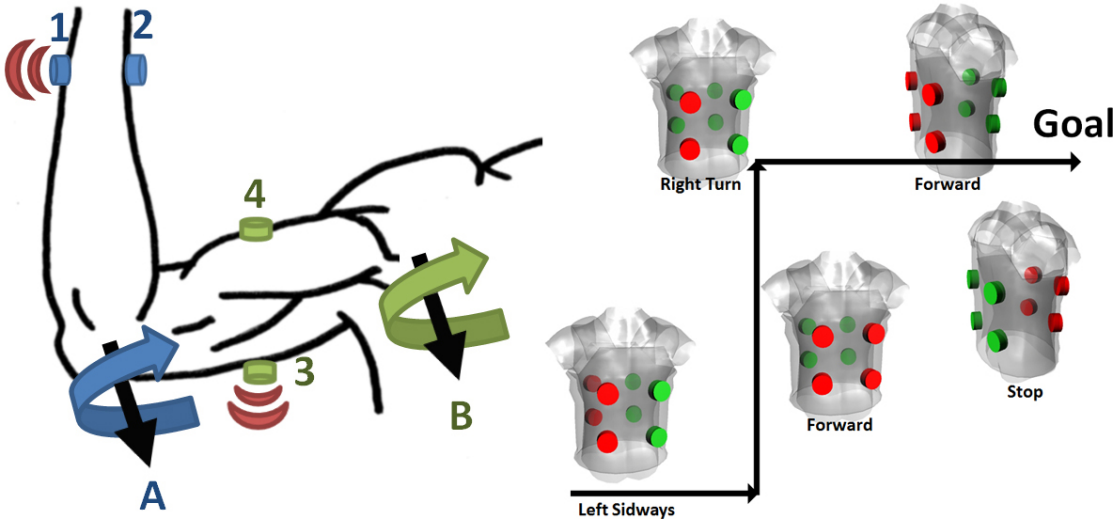


Figure 6.9: (Left) Vibrotactors 1-2 and 3-4, indicate rotations around axes A and B, respectively. (Right) Activation patterns for navigation with pneumatic feedback vest (active tactors = red).

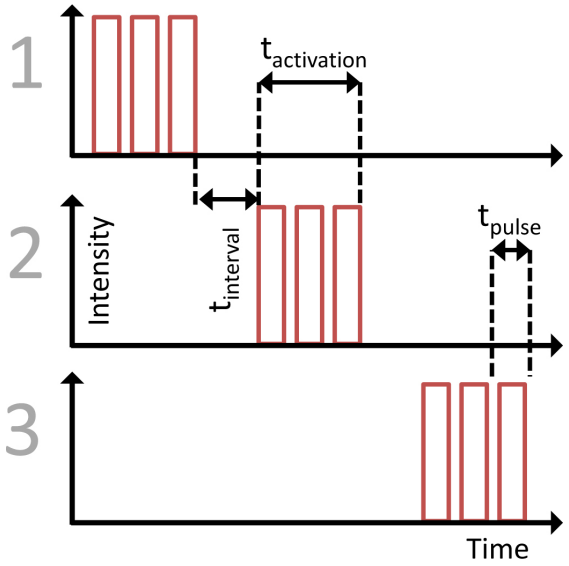


Figure 6.10: Sequential pulsing of three vibrotactors to indicate directional speed.

approximately two seconds, which was chosen as a practical maximum for our applications. $t_{activation}$ is the sum of pulses of a single tactor. The actuation intervals are calculated from user anatomy (i.e., arm length) and target velocity (v_{target}) using the equations:

$$t_{interval} = \frac{\frac{armlength}{v_{target}} - (3 * t_{activation})}{2} \tag{6.1}$$

$$t_{interval,calibrated} = t_{interval} * factor_{calib} \tag{6.2}$$

Preliminary experiments with five study participants on perceived absolute speed, indicate individual differences that can be corrected with a calibration factor ($factor_{calib}$) as described in Section 6.9.3.4.

The module implementing the rich vibrotactile display can also be configured to present translational forearm-directions through sequentially triggered tactors in seven different directions (See Figure 6.11). It allows the guidance towards target poses, where target speed can be chosen according to factors such as desired loop time or optimal user perception.

Implementation Figure 6.12 shows a somewhat simplified class hierarchy of the haptic feedback module, with the most important classes. *FeedbackCalc* is the base class for the three main methods of haptic output specified above and can be attached to *GameObjects*. Specialized classes then implement the specific method. *FeedbackCalcColide3D* calculates direction and intensity of output based on a physics simulation and collisions of objects. *FeedbackCalc(Rot/Pos)3D* calculate feedback based on positional/orientation differences between *GameObjects*. *FeedbackCalcVel* computes a representation of directional velocities, which has been used for example in the preliminary study described in Section 6.9.3.4.

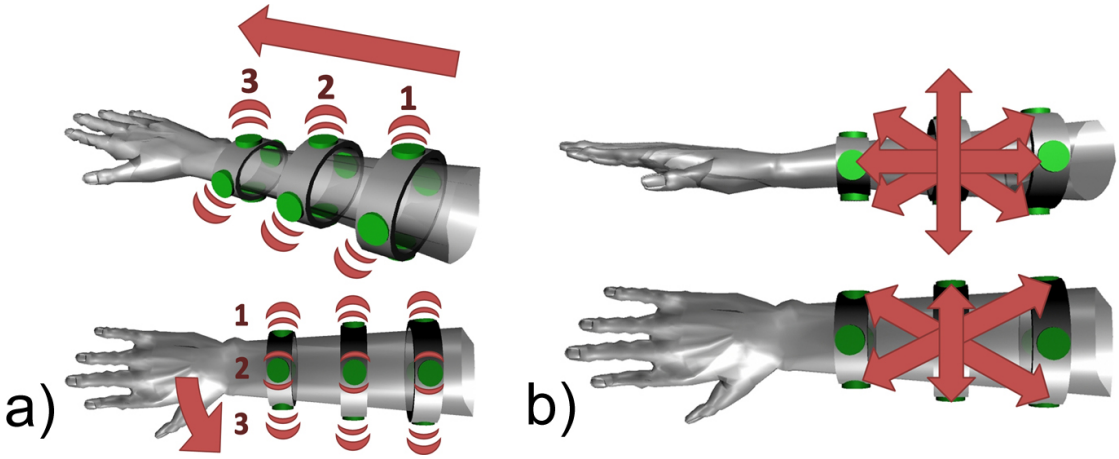


Figure 6.11: Sequential triggering of vibrotactors can be used to indicate a) speed and b) direction vectors.

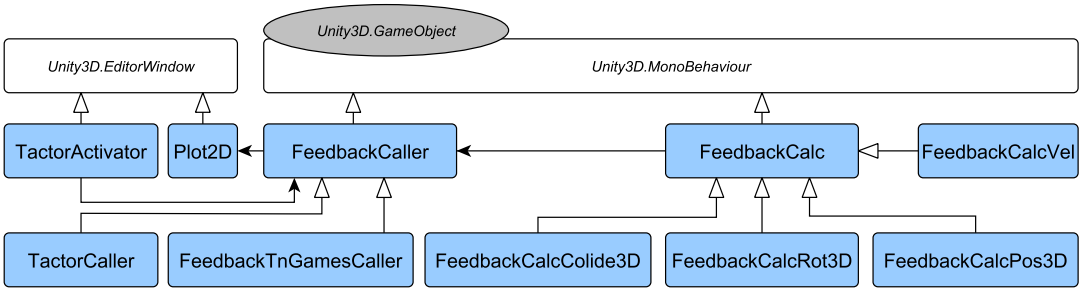


Figure 6.12: Haptic module class hierarchy.

FeedbackCaller is the central class to the module. It provides an interface that other classes and scripts can use to initiate haptic output with specified strength in a direction (Note that all three methods above provide feedback in a certain direction). Classes inheriting from *FeedbackCaller* like *TactorCaller* and *FeedbackTnGamesCaller* provide implementations of specific haptic devices (Tactaid vibrotactile actuators and TN Games 3RD Space vest) as described in Section 6.6.6. Devices themselves can be accessed through the plug-in mechanism of Unity3D, while we provide a common interface to the application.

To make authoring of applications with haptic feedback easier, we have added a few extensions to the Unity3D editor. The most important are *TactorActivator* and *Plot2D*. *TactorActivator* provides a GUI for adding and configuring haptic feedback to *GameObjects* as shown in Figure 6.13. Our GUI makes it straightforward to visually map the feedback to different limbs of a virtual avatar and to arbitrary feedback axes on the body. *Plot2D* on the other hand creates a window, which can display haptic feedback exerted on arbitrary *GameObjects* at runtime (Figure 6.14). Both access *FeedbackCaller* to set/retrieve parameters.

Directional speed can only be triggered serially as shown in Figure 6.11, through sequential

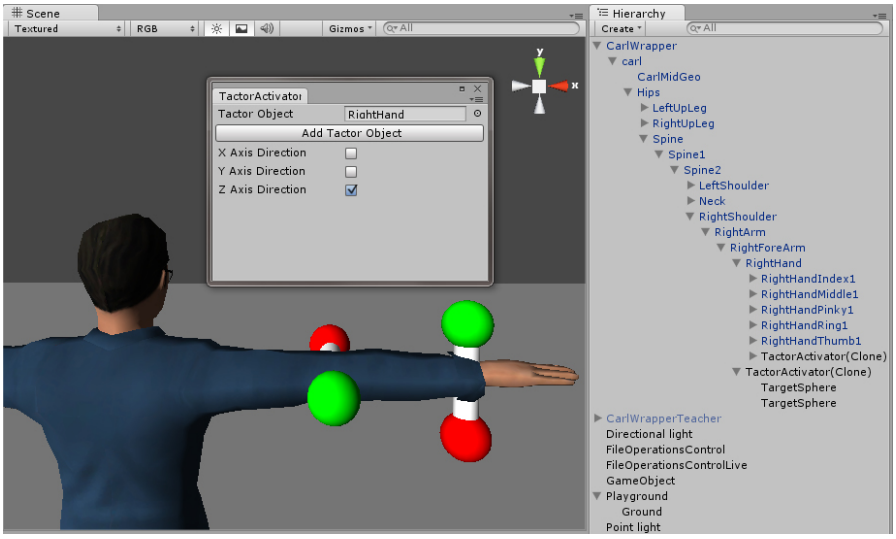


Figure 6.13: Our graphical interface allows direct editing of haptic feedback on an avatar in a virtual 3D environment.

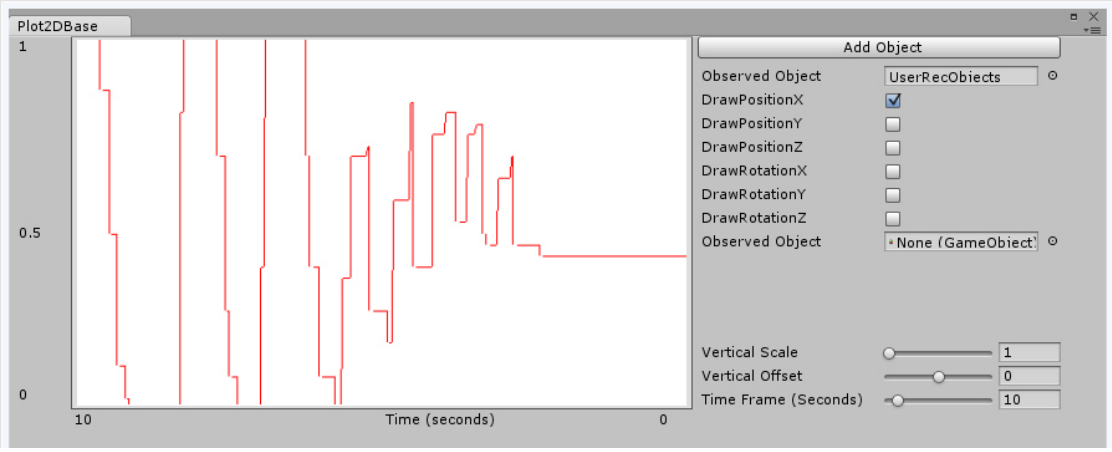


Figure 6.14: Our GUI provides straightforward access for controlling and adjusting tactile feedback.

tactor activation, and depends on a presented prerecorded teacher movement. In the current implementation the sequences are independent of the user's performance and can be considered support rather than feedback.

6.6 Devices and Integration

As introduced in Section 6.3 I/O devices are one of the core components of a VR/AR system. We have integrated support for several tracking systems in our framework to allow interaction

with the whole body. Furthermore, new software components for biosignal amplifiers allow for acquisition of physiological data. Finally, the ARTiFICe module for haptic output devices provide different options for haptic feedback. Currently, ARTiFICe supports the following input and output devices:

- Iotracker via customized interface or VRPN
- Microsoft Kinect for Windows
- TMSi Mobi
- g.tec g.MOBIlab
- ARToolkit markers
- 3D Connexion SpaceNavigator
- Razer Hydra
- Optical tracking targets (iotracker, wide area)
- (Handheld devices - Android 2.1 or higher)
- Vibrotactile feedback (actuators from e.g. Tactaid)
- Pneumatic feedback (3RD Space Vest from TN Games)
- All devices that are supported by OpenTracker and VRPN

The following sections provide a categorization of input devices and details on their integration.

6.6.1 Desktop Interfaces

Design For desktop setups, ARToolKit [65] as well as ARToolkit+ [158] are easy to use tracking libraries providing a square planar shape for pose estimation and an embedded 2D pattern for distinguishing markers. They calculate camera position and orientation relative to physical markers in real time and thereby enable the development of a wide range of Augmented Reality applications. ARToolkit is usually applied for desktop based AR environments while ARToolkit+ enhances the original ARToolkit library and is optimized for usage on mobile devices.

Implementation We use ARToolkit+ within our framework, which has been previously integrated into OT by implementing the *ARToolKitSource* node. OpenVideo [101], a data integration- and processing framework, is applied to acquire video frames from the webcam, which are processed by ARToolkit+ and later streamed into Unity3D to provide a view of the real world scene.

We integrated a 3D Connexion SpaceNavigator (3D mouse) [1] and Razer Hydra [60] into OT. We have implemented the OT source nodes *SpaceDeviceSource* and *HydraSource* to provide easy to use 6DOF desktop interaction. For an in depth discussion of the implementation and about working with ARTiFICe and Razer Hydra, the reader is kindly referred to [32].

6.6.2 Tracking Devices

Design Many applications especially with semi as well as fully immersive setups require tracking beyond the desktop. Therefore, iotracker [105] as a infrared optical tracking system has been integrated. It tracks arbitrary physical objects equipped with passive markers with an update rate of $60Hz$ and very low latency and jitter and high accuracy.

Implementation We integrated iotracker in our framework by interfacing OT using VRPN and through the OT *UnitySink*-node as introduced in Section 2.8. A simple *EventGenerator* inserts events of the OpenTracker standard data type into the tracker tree. This is straight forward, since that is what OpenTracker has originally been designed for. Therefore, the iotracker server software works as a VRPN server and a *VRPNSource*-node is used to connect OpenTracker to the iotracker.

6.6.3 Motion Capture

Design The MoCap systems described in Chapters 4 and 5 are well integrated with the ARTiFICe framework. Depending on the application, it might be desirable to use either the MoCap data of the complete skeleton (e.g. for animation of an avatar) or only specific bones or positions on the skeleton (e.g. hand positions for selection and manipulation tasks). For a straightforward use of bone poses we have implemented the interface of the 3D Event Handling Module described in Subsection 6.5.2.3. This results in more information than necessary to be transmitted since all limbs are linked together and therefore positions are not arbitrary. However, the implementation is straight-forward, since it only requires minor extensions to the existing 3D Event Handling Module. For transmission of the whole skeleton transformation in consistent form we used a custom network interface.

Implementation For streaming skeleton-based motion-capture data into OpenTracker, we have implemented the following two options:

1. The MoCap software interprets every bone as single rigid-body target and makes it available over VRPN. Accordingly, the OT *VRPNSource* node produces events with all the orientations and positions. This interface is implemented in the iotracker server software as well as in the software of the Kinect based wide area tracking system detailed in Chapter 5.

2. Before the actual MoCap data, the skeleton model is transmitted to the framework. After that only a pivot point and the joint angles are propagated into the data flow network through a specialized OT source node (*IOTMCSource*) node. The framework and dedicated OpenTracker nodes can then use the skeleton model, to reconstruct the pose starting from the pivot point and following the joints angles. Therefore, the required data rate is reduced by almost half compared to transmitting the whole pose for each bone. This option can be considered minimal in terms of the required data. Efficient transmission can be important in case the Tracking-PC has to transfer the tracking data over a network.

This representation also makes animation easier than with the other option. This option is only available in the iotracker software.

Besides marker-based optical tracking we also wanted to allow markerless full body motion tracking. Therefore, we integrated Microsoft Kinect using the OpenNI/NITE [110, 109] framework and FAAST [143]. OpenNI/NITE provides an API to access raw depth data as well skeleton data, which are calculated based on the depth data. FAAST runs as self-contained application and reads this data. It provides gesture recognition support and full body tracking data via VRPN (see Section 2.8 for details) to OT. Using the OT *UnitySink*-node introduced in Section 6.5.1.2 and method 1 described above, we feed real-time skeleton tracking and gestures into the ARTiFICe core. Data from the wide area MoCap solution described in Section 5 can be integrated through VRPN in a similar fashion.

For both options, there is also the possibility of identifying certain moves or postures in OpenTracker using scripts. These in turn can then be used to trigger e.g. button events and propagate only those through the network. Additionally, for option 2, the *IOTMCSkelMod* OT filter node calculates "points of interest" and derived parameters like walking or movement velocities for use in the application layer as described in Section 4.3.3.

6.6.4 Biosignal Acquisition Systems

Design Muscle activity not always results in visible motion and therefore, it is difficult to track using conventional MoCap devices. Furthermore, it has been shown that feedback supported with biosignals, muscular activity in particular, is a suitable tool to reduce disabilities of patients with chronic musculoskeletal pain [157]. Other physiological measurements like heart rate or skin conductance can also be highly relevant in a rehabilitation scenario [47].

Therefore, we have integrated two biosignal acquisition devices (TMSI Mobi [149] and Gtec g.MOBIlab [50]) into our system through the middleware layer described in Section 6.5.1. Both biosignal acquisition systems have a number of sensors including electroencephalography (EEG), electrocardiogram (ECG), electromyography (EMG), galvanic skin response (GSR), respiration and others. For our purposes in chronic pain rehabilitation we are primarily using EMG as shown in Figure 6.15, while other sensors would be easy to activate.

Implementation As introduced in Sections 2.8 and 6.5.1 OpenTracker is a generic data-flow network designed to deal with tracking data, but not limited to it. Therefore, we chose OpenTracker as an abstraction layer that provides a common interface to all input devices and biosignal sensors used throughout our framework. Two biosignal acquisition devices have been integrated in OT, the gMOBIlab of g.tec and the Mobi produced by TMSi. Both devices have dedicated OT modules, which provide and run a source node in the OT data flow network. The source node in turn generates events from the data received from the sensors. The modules can be configured with associated elements in the OT configuration XML file (see Section 2.8 and [114, 100] for details) and run a separate thread for data acquisition.

The OpenTracker integration of gMOBIlab accesses gMOBIlab via the serial port and is largely based on the implementation of Alexander Bornik as contained in [100]. Several adaptations have been made especially in the *MobilabDriver* component to add support for Windows

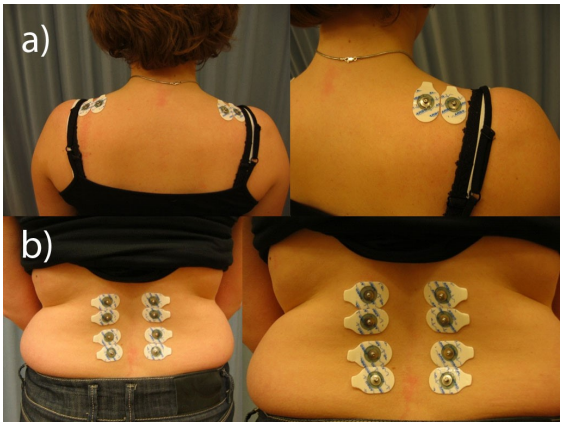


Figure 6.15: a) EMG-electrode placement for upper Trapezius muscle. b) EMG-electrode placement for the lower back.

and the *MobilabSource* OT-node to improve convenience. *MobilabDriver* provides methods for opening and closing the serial port and retrieving data/sending commands to gMOBIlab. It therefore encapsulates the whole gMOBIlab and serial communication code which is used for accessing the device. The data provided to connected filter nodes is the original gMOBIlab data at the full sample rate without any transformations. These operations can be done within OpenTracker using filter nodes e.g. for smoothing EMG data.

The application described in Chapter 7 required a higher sampling rate for the EMG data and therefore the TMSi Mobi featuring sampling rates of up to 2048 Hz has also been integrated into ARTiFICe/OT.

The integration of the TMSi Mobi was implemented in a similar fashion to the gMOBIlab. *TMSIDriver* handles all the communication over serial port/Bluetooth. However, the communication protocol is more complex than with the gMOBIlab and data packages are compressed for transport over Bluetooth. Therefore, *TMSIDriver* and related components have to also take care of message parsing and decompression. *TMSISource* passes samples with the raw data on to the data flow network, where processing of the data can be similarly handled as with gMOBIlab.

In addition to the basic gMOBIlab and Mobi device integration, a filter has been implemented, which can extract the heart rate from the ECG signal. *HeartrateFilter* implements an OpenTracker filter node looking for local maxima/minima in the ECG signal and using the timestamps of the events averages the heart rate over a configurable amount of time. EMG data of the biosignal acquisition systems is also processed using specifically designed OpenTracker filters. Typical EMG data ranges between 6 and 500 Hz. To avoid heavy movement and contact artifacts the sampled signal is typically sent through a high pass filter [73]. Therefore, we have implemented a custom configurable high pass filter for OpenTracker. Raw EMG spikes have random shape, due to the nature with which different motor units in the muscle fire during movement, which is why usually smoothing algorithms are applied [73]. In our case we have decided for a Root Mean Square (RMS) filter configured to average over 128 samples. These settings have been shown to work well in previous research on chronic neck and shoulder pain

[20]. The filtered data is finally sent to the *UnitySink* node, which connects to the application layer.

6.6.5 Video and Audio Output

For rendering in the ARTiFICe framework we are using Unit3D's pipeline [151]. Unity3D's rendering engine supports both OpenGL and DirectX. A flexible camera management allows to render stereo output for different output devices in semi-immersive and immersive setups. For Augmented reality applications the camera image can be used as an OpenGL texture and rendered in the background. Furthermore, Unity3D's shader system allows to efficiently correct for lens distortion in HMDs with wide fields of view. Finally, recent updates have optimized the rendering pipeline for virtual and augmented reality devices natively supporting several HMDs.

Unity3D's audio system supports 2D and 3D audio, which can be spread out between speakers (stereo to 7.1 surround) [151]. Furthermore, a number of blending effects and filters are available.

6.6.6 Haptic Feedback

As introduced in Section 6.5.2.7 the haptic feedback module in the application layer can implement classes inheriting from *FeedbackCaller* like *TactorCaller* and *FeedbackTnGamesCaller*. that wrap the drivers of specific haptic devices (Tactaid vibrotactile actuators and TN Games 3RD Space vest). The drivers are accessed through the plug-in mechanism of Unity3D, while we provide a common interface to the application.

Vibrotactile Feedback Vibrotactile feedback is provided using tactors from Tactaid. Our dense tactile display uses twelve vibrotactors on the arm, as shown in Figure 6.16. Our custom control board [41] drives the tactors at 250 Hz, the recommended frequency for human skin perception [38], and communicates wirelessly with the host PC using Bluetooth. We use pulsed vibrations, instead of continuous, as it is better for perception, as shown in related work [139] and confirmed in our early experiments. We have experimented with different activation patterns and tactor configurations. Figure 6.11 shows the positions where vibrotactile stimuli are applied for our dense tactile display.

Pneumatic Feedback Pneumatic feedback is provided using a 3RD Space Vest from TN Games [43], which applies pressure using four actuators on the chest and four on the back, as shown in Figure 6.9. Designed as a gaming peripheral the eight pneumatic actuators driven by a small compressor can be controlled separately through valves controllable through TN Game's SDK and driver.

6.7 Application Development Workflow

In the following we briefly describe the workflow to build a new VR/AR application. First, a new Unity3D project is created and the ARTiFICe framework is added to the project folder. All

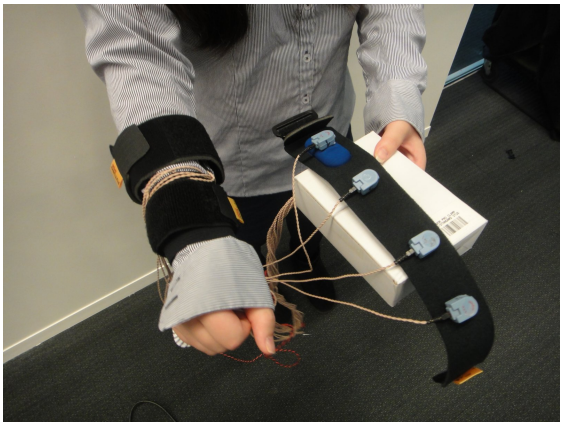


Figure 6.16: Vibrotactor hardware.

input devices and biosignal acquisition systems are configured in a single OT configuration file. Cameras, lights and scene objects are then added to the virtual environment using the Unity3D graphical scene management. They are encapsulated as Unity3D *GameObjects*. Virtual entities which act as tracker-, interaction- or selectable scene objects are subsequently connected to the according classes of the ARTiFICe framework. Finally, the project is built and deployed to the desired platform and run as single or multi user VR/AR application.

By focusing on a well-defined virtual scene management, loose coupling of input devices and interaction techniques, we created an environment which allows technically experienced users to adapt the framework to their needs for application development. In addition, we provide an easy to use framework to help students getting over the initial hurdles of creating quick prototypes of an embodied AR experience.

6.8 Interfacing with the Framework from other Tools

The ARTiFICe framework is also flexible and extensible in terms of its interfaces being open for access by applications or modules outside the framework. Therefore, we used for example the socket interfaces with our customized XML protocol described in 6.5.2.3 for medical evaluation (C-Motion Visual3D) for character animation (Autodesk MotionBuilder). For both tools we have implemented plug-ins capable of utilizing the data. Furthermore, other tools supporting VRPN [147] can be easily interfaced through the interface introduced in Section 2.8 and 6.5.

6.8.1 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.

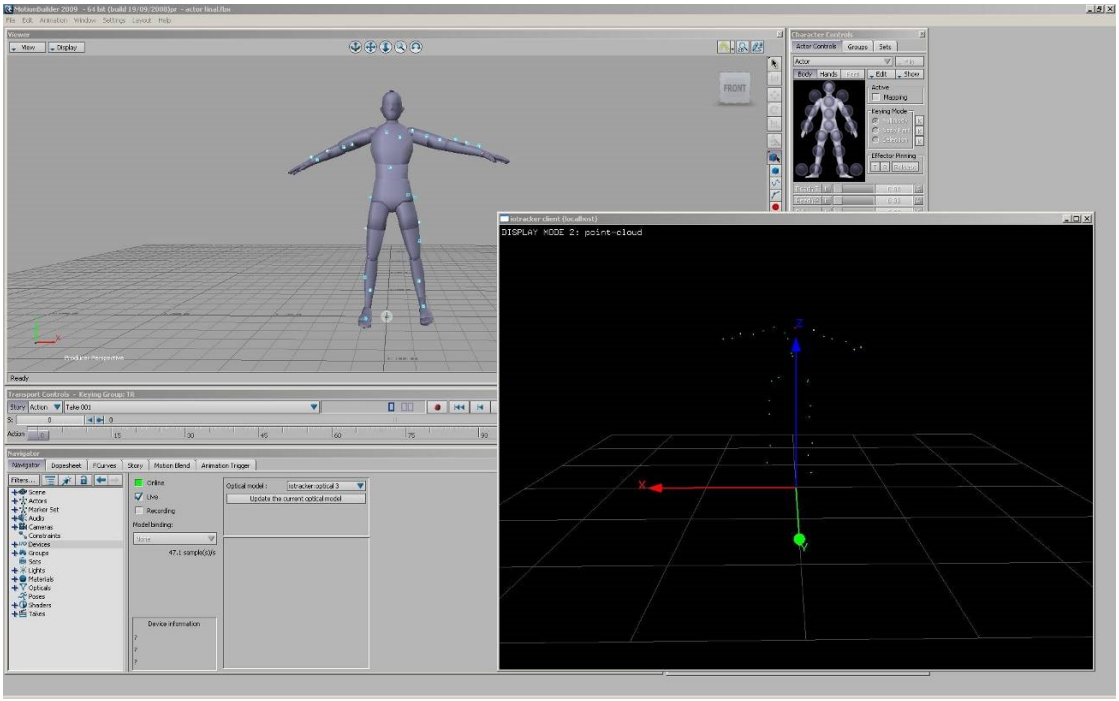


Figure 6.17: MoCap data is captured using a MoCap suit and the iotracker (visualized in the lower right corner using the iotracker-client). The data (a labeled point cloud) is then streamed to MotionBuilder, where it can be attached to a character. The pose of the character is then adapted to the tracking data in real time.

6.8.1.1 Integration with MotionBuilder

The basic concept of integrating MoCap data from the iotracker into MotionBuilder is shown in Figure 6.17. The iotracker sends the data to a specified client workstation using its own XML-protocol. 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 so-called 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.

The MotionBuilder integration works with rigid body targets as well as a MoCap suits. In order for the MotionBuilder plug-in to work, the marker-labels have to stay constant during the whole session. Therefore, for MoCap data we have used POIs as virtual markers instead of the real physical markers, which can't be identified per-se. 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. Visual inspection of the resulting animation showed good compliance.

Due to the use of POIs 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 even without the use of rigid-body targets.

6.8.2 C-Motion Visual3D

Visual3D offers biomechanical analysis for researchers and clinicians. Clinically, the software package is used for performance-analysis and movement-assessments.

Visual3D supports any arbitrary marker sets, conventional gait, Helen Hayes and variations, custom marker configurations, and minimal marker sets. Additionally, it supports a full 6 degree of freedom segmented analysis. It provides kinematics, inverse dynamics (kinetics), inverse kinematics (global optimization) and even some forward dynamic computations. Furthermore, computations for analyses can be added in Visual3D or with MatLab.

The Visual3D software processes synchronized analog data (EMG, force plates, etc.) and the data from motion capture in real-time. Therefore, Visual3D can be used to collect data and provide real-time feedback of selected movement activities. This in turn enables the therapist to gain analysis data during the therapy sessions.

Finally, computations and calculations in Visual3D are well documented with little proprietary functionality - public references exist for many calculation performed. There are numerous filters available for signal processing, and several automatic event recognition methods are supported.

6.8.2.1 Integration with Visual3D

Integration of the ARTiFICe 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 6.18. First a customized plug-in connects to the tracking software (directly or indirectly via OpenTracker) using TCP-sockets. The framework then sends XML-packets to the plug-in containing labeled POIs.

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.

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 6.18).

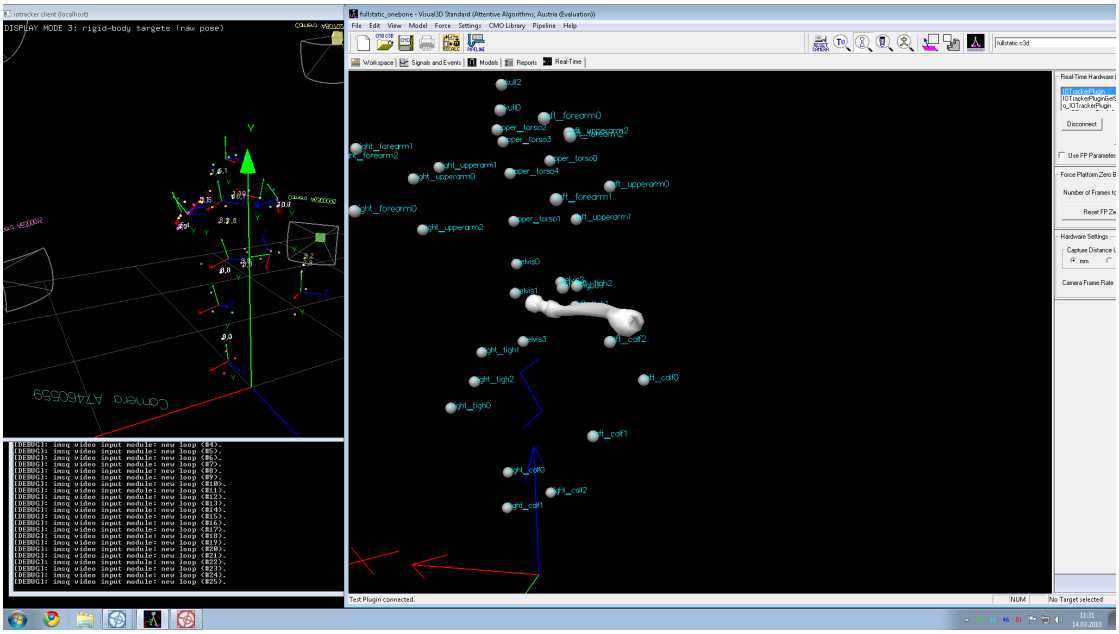


Figure 6.18: Screenshot of the iotracker-client and Visual3D showing visualizations of the local bone-coordinate-systems as well as the POIs with their labels. The thigh-bone shows exemplary how a bone is scaled to the POIs and animated live by using POI-positions.

The pose for each frame is updated using the inverse kinematic algorithms provided by Visual3D, which uses 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. C3D is an open file format created by Vicon Motion Systems 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.

6.8.2.2 Evaluation of Motion Data Using Visual3D

Visual3D offers functionalities to create graphs and charts to evaluate the patient's movements.

The therapist starts the iotracker-server as well as Visual3D. In Visual3D, she then loads the skeleton-model predefined for the virtual marker-set being used. Then she initializes the plug-in (as depicted in Figure 6.18), which connects to the framework and starts streaming the data.

Using the "Model Builder Real-time" feature from Visual3D, she 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 her 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.

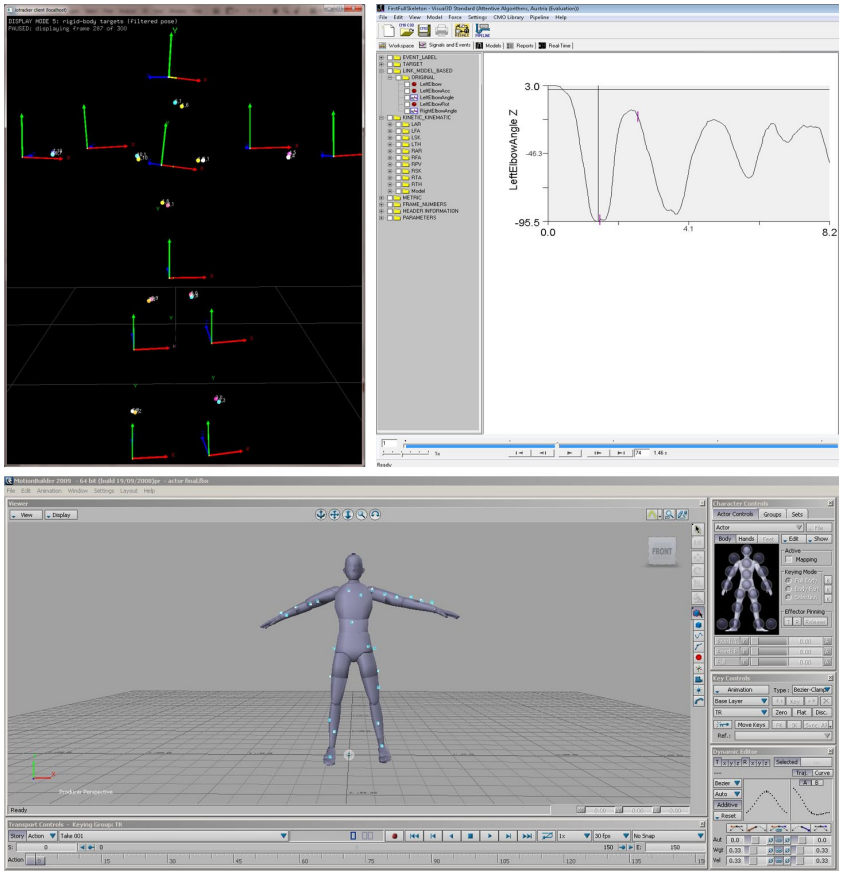


Figure 6.19: Visualization of the skeleton model's local coordinate systems (upper left), joint angle curve after the data was streamed into Visual3D (upper right), Motion Builder (bottom) showing animated avatar with marker positions.

6.9 Results

Our ARTiFICe framework was heavily tested and evaluated using different setups, as described in detail in Sections 6.9.2 to 6.9.3.5 and Chapter 7.

6.9.1 Test Platform

The framework was tested on various workstations, mostly running Windows XP and Windows 7 (32/64bit). Many parts of the framework, can also be deployed on Mac OS X. Mobile applications were run on multiple Android devices, all running Android v2.1 and higher.

6.9.2 Test Environment

ARTiFICe was used for the master's degree course "Virtual Reality Lab Exercise" at Vienna University of Technology from winter term 2011/12 on up till now. In total, more than 150 stu-

dents developed distributed and collaborative VR/AR applications with ARTiFICe, using several interaction techniques in combination with ARToolkit markers, 3D Connexion SpaceNavigator and Microsoft Kinect for Windows.

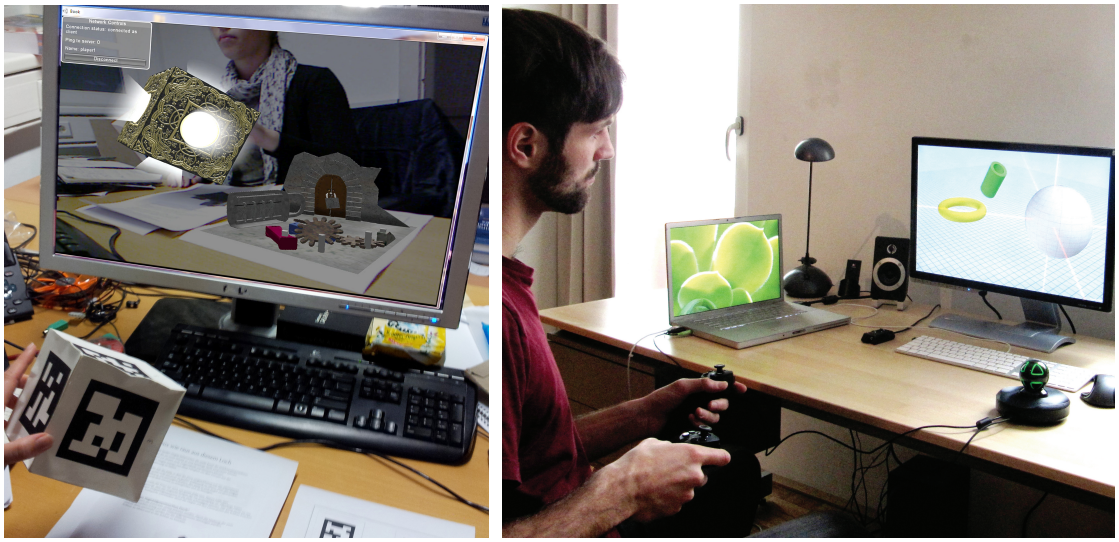
ARTiFICe is currently used within a number of on-going research projects which involve fully immersive setups, distributed mobile interaction and various input devices.

6.9.3 VR/AR Setups

In the following section, we present different VR/AR environments that were developed with ARTiFICe using various hardware setups and interaction devices. These systems are using a wide variety of devices and offer different degrees of immersion as introduced in Section 2.7.

6.9.3.1 Singe-User Non-Immersive VR/AR

A desktop VR application was realized using Razer Hydra as a highly accurate 6DOF interaction device. In an application for geometry education virtual scene objects are controlled using the Hydra, as illustrated in Figure 6.20b). A desktop AR application using multiple ARToolkit markers forming a MagicBook [14] is shown in Figure 6.20a). An ARToolkit cube is used as multiple-purpose interaction device. A serious game targeting rehabilitation in the context of subacromial impingement syndrome has been developed [39] (see Figure 6.21). The game utilizes the Microsoft Kinect for full body motion capture of the patient, features multiple fully customizable exercises and embeds them into meaningful play for the patient. Gestures of the unencumbered arm are used for general interaction with the game environment, while movement of the disabled arm is only required during specific exercise interactions. The game is visualized on a 2D projection screen.



(a) Desktop AR setup with ARToolkit (b) Desktop VR setup with Razer Hydra

Figure 6.20: Single-user desktop VR/AR setups.

Figure 6.21: VR setup with Kinect full body MoCap for interaction with a serious game for rehabilitation.

6.9.3.2 Multi-User Non- & Semi-Immersive VR/AR

In addition to non-immersive VR/AR desktop setups, we used ARTiFICe to realize combined non/semi-immersive environments. In Figure 6.22, a collaborative and distributed multi-user scenario is shown. User 1 controls the speed and direction of a flying object by gesture recognition and full body motion capture using Microsoft Kinect (Figure 6.22a). User 2 interacts with a 3D Spacenavigator, controlling the height of the flight and clearing the object's flight path using GoGo interaction technique (Figure 6.22b).

User 2 uses a non-immersive 2D screen visualization in a desktop environment, while User 1 is provided with a 3D visualization, using stereo projection with shutter glasses.

6.9.3.3 Multi-User Semi- & Fully Immersive

We used our framework to develop a server-client application to provide training for prosthesis patients [96]. The software consists of a server application to control all parameters and a client application to visualize the virtual environment in the HMD. In Figure 6.23, a demo setup of this fully immersive application is shown; a 6DOF rigid body target is used to track the user's arm for controlling the virtual prosthesis. Tracking of HMD- and arm target is done using iotracker. Customized electromyography sensors and amplifiers are used to control the opening and closing of the virtual prosthesis.

Stereo projection provides the HMD view to an audience to share the HMD user's experience for discussion and explanations in a semi-immersive VR/AR.

6.9.3.4 Haptic Feedback Module Preliminary Experiments

We conducted a small preliminary evaluation of ARTiFICe's haptic feedback module. We built a small application on top of the framework to assess perceptional response of participants to vibration patterns for arm speed indication. Five subjects volunteered for the user study. Two of the participants (1 and 2) were male and three (3-5) were female. Two movements (Figure 6.11), intended for translating and bending/extending an arm, were tested. The system randomly provides several patterns, which show different speeds. The indicated speed is normalized by

(a) Stereo projection with full body motion capture.

(b) Desktop VR with 3D Spacenavigator.

Figure 6.22: Multi-user desktop/semi-immersive VR setup.

Figure 6.23: Immersive VR using optical tracking, combined with multi-user semi-immersive stereo projection.

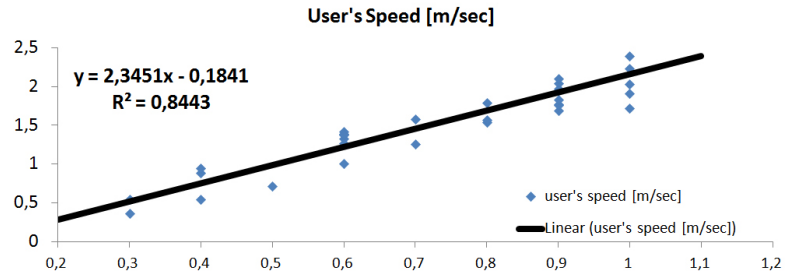


Figure 6.24: Plot of a user's measured arm speed versus the indicated arm speed

dividing an user's arm length by burst duration as shown in Figure 6.10. Users are instructed that the speed at which the vibrotactile stroke moves is the same as the required arm movement speed and that they should follow the movement as indicated. For the larger part of the experiment, the subjects were wearing headphones producing pink noise in order to avoid the subjects to be affected by sounds from vibration motors. The application recorded the movements as captured by iotracker. Figure 6.24 exemplarily plots the measured speed (vertical axis) at indicated speeds (horizontal axis) of one subject. It can be observed that the standard deviation increases with the indicated speed, which could mean that higher speeds are harder to perceive or to reproduce. However, this has to be evaluated in further studies.

We fitted regression lines to the data and performed an analysis (values Table 1). The values of proportion of variance explained (R^2) have a median of 84% for translational and 87% for rotational tests. Furthermore, large F values (Table 6.1) indicate high significance of our linear regression model. Therefore, a **strong linear correlation** between the indicated speed and the measured speed can be observed.

The only exception is translational indication with subject 5. This might be due to an insufficient instruction on the experiment. From the collected data, we could see that subject 5 translated the instructed speed into the frequency with which the arm was moved and not into the actual speed, which was kept constant. Thus, in this case the regression analysis shows no correlation between indicated speed and subject reaction speed.

Subject	Rotation R^2	Rotation F value	Translation R^2	Translation F value
1	82%	193	88%	164
2	87%	141	87%	71
3	88%	254	84%	223
4	87%	237	64%	59
5	77%	108	4%	1,45

Table 6.1: Variability and significance of regression

From the strong linear correlation of the other subjects, however, we can interpret, that different speeds can be perceived and transformed into actual movement speeds. The proportionality coefficients are unequal to one, which means that the subjects were able to reproduce a speed linearily depending on the indicated speed, but not the actual absolute speed. We hypothesize that this can be improved by calibration which maps the actual speed to human reaction. We scaled the indicated speed linearly relative to the calculated coefficients and continued to apply a linear model.

After **calibration** using the correlation from the first part of the study, the users repeat the same experiment to see if the system is able to show absolute speed. One user finished the evaluation after the calibration. In translational movement, regression analysis results showed the proportionality coefficient improved from 0.79 to 0.89. In rotational movement, the proportionality coefficient improved from 4.6 to 1.6. This result supports our idea that, after calibration, the system can indicate absolute speed for a user.

6.9.3.5 ARTiFICe as Basis for a Motion Guidance System

Based on our ARTiFICe framework we have designed and implemented a Motion Guidance System based on visual and tactile feedback, with a number of modules providing different modalities and dimensions of feedback. The components in our system can be flexibly combined into new applications thanks to their modular nature. Our components generate visual, vibrotactile and pneumatic feedback for generic use with arbitrary limbs, or for specific tasks like arm movement or navigation as described in [126]. These methods for motion guidance could be also interesting for motor learning in a rehabilitation scenario.

6.9.4 Conclusion and Future Work

This chapter provided details on our VR/AR framework ARTiFICe with a view towards use with serious games in rehabilitation. Many applications and games have been implemented based on ARTiFICe. Various devices have already been integrated in the framework and new ones will be added in the future. Furthermore, we plan to provide our framework as open source project to the developers and research community.

CHAPTER 7

A Serious Game for Chronic Pain Rehabilitation

7.1 Introduction

This chapter introduces the medical background, design, development and evaluation of a serious game prototype intended to assist rehabilitation of patients with chronic pain of the lower back and the neck. The findings presented here have been published in [129, 131, 128, 63]. Experts in movement science from Roessingh Research and Development (RRD) helped to de-

Table 7.1: Mini games, their medical rationale and the MoCap data used.

Game name	**Game description**	**Medical rationale**	**MoCap data**
Temple of Magupta	The player runs through an ancient temple, collects artifacts and avoids obstacles	Physical reconditioning, increase walking speed	Movement rate
Face of Chronos	The patient climbs a mountain by extending the arms upwards until the next hold is reached and collects artifacts placed on holds	Increase reaching ability, velocity and smoothness of the motion, relaxation of the trapezius muscle after reaching	Movement characteristics of the hand and arm (path, velocity), muscle activity of the left and the right trapezius muscles (EMG)
Three Wind Gods	The player imitates a series of head movements executed by fictive characters.	Increase Cervical Range Of Motion (CROM), and increase velocity and smoothness of cervical motion	Measures of CROM and current rotation (flexion/extension, right/left bending, left/right rotation), velocity, acceleration

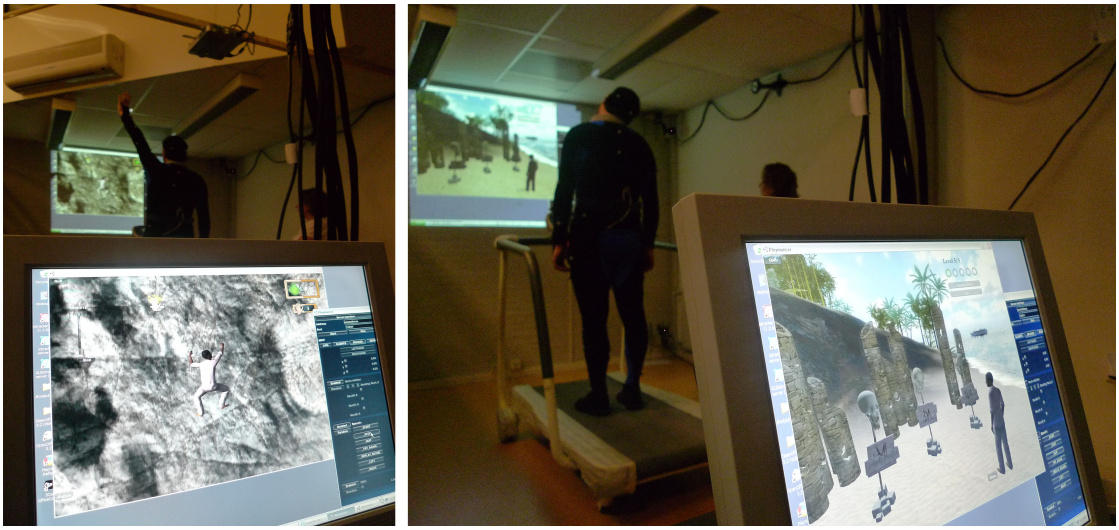


Figure 7.1: Our MoCap system and two of the mini games during the preliminary testing. “Face of Chronos” (left), “Three Wind Gods” (right).

termine the medical requirements and led efforts on the medical evaluation during the EU-FP7 project PlayMancer (FP7-ICT-215839-2007). RRD is a research center for rehabilitation technology and linked to the Roessingh rehabilitation center in Enschede. The game has been designed based on the considerations and principles introduced in Section 2.9 and consists of a common game environment into which multiple mini-games containing different rehabilitation exercises are embedded. An overview of the mini games, their medical rationale, therapy goals and used MoCap data is listed in table 7.1. They are embedded within an adventure setting, linking the games with a common story line and the successful concept of exploration games. Pictures of test users playing the mini games can be seen in Figure 7.1. All mini games provide the patient with visual and textual feedback on his performance during gameplay. Furthermore, the player receives virtual collectible items and a narrative reward in the form of a story.

Technically, the game is based on the ARTiFICe framework as introduced in Chapter 6. The full body MoCap system as described in Chapter 4 and the biosignal acquisition system introduced in Section 6.6.4 are employed to track exercises and work as main source of input data for the serious game.

The following sections describe:

- Medical background of chronic pain (Section 7.2).
- Game design and how general and specific requirements are met (Section 7.3).
- Evaluation at multiple stages of development (Section 7.4).

7.2 Medical Background & Requirements Specific to Pain Rehabilitation

According to cognitive-behavioral models [52, 155], patients with chronic pain are in a vicious circle. Deviating activity patterns, like maladaptive pain-related cognitions (i.e. fear of movement), inadequate coping strategies (i.e. avoiding physical activities) and physical disuse result in a decrease in physical condition, muscle strength and increase in illness and sick-role behavior [52, 152]. Therefore, therapeutic activities performed in pain rehabilitation practice, focus on adapting cognitions, improving coping strategies, increasing physical functioning and normalization of activities of daily living. An important way to do this is by letting patients experience and execute physical exercises. Exercises commonly used in chronic pain rehabilitation focus on:

- Mobility and coordination by activities that target increased range of motion, increased velocity and smoothness of the motion.
- Improving a patient's physical conditioning by endurance exercises like walking.
- Relaxation of muscles by exercising both total body relaxation as well as local muscle relaxation.

Changing motor functioning is considered a dynamic process. Subjects need to become aware of their inadequate functioning during daily life and need to learn skills to change it. Subsequently, they need to be motivated to develop intentions to change and then actually change. So, besides monitoring physical performance, giving appropriate feedback about the performance to the user is important in the context of pain-rehabilitation and thus also for serious games in this context.

7.2.1 Mobility and Coordination

Sandlund et al. showed that patients with neck-/shoulder pain have reduced ability to reach overhead due to their pain [120]. Besides, patients with neck-/shoulder pain demonstrate decreased range of motion (ROM) and lower peak velocity compared to asymptomatic controls [163, 137]. Furthermore, patients exhibit significantly higher variability in ROM and reduced smoothness of movements. Therefore, many patients can significantly benefit from increasing their neck mobility by exercising.

7.2.2 Physical Reconditioning

Previous studies have shown that patients with chronic low back pain (CLBP) have lower preferred walking velocity compared to controls [140] and their physical activity is deteriorated due to disuse and deconditioning [53]. Little by little these patients become less active, physically and socially. By walking, patients with chronic pain can, train their physical condition and increase their walking velocity. Furthermore, the walking capabilities of a patient can also be used as a measurement of physical condition through e.g. the six-minute walking test [8].

7.2.3 Relaxation

Relaxation of the whole body as well as local muscle groups is an important part of a healthy activity pattern [152]. In this work we concentrate on local muscles or more specifically the upper trapezius. Changes in the activation of upper trapezius have been observed in people with neck and shoulder disorder [20]. The electromyography (EMG) of patients with neck pain demonstrated greater activation of accessory neck muscles during a repetitive upper limb task compared to asymptomatic controls. During separate elevation of the shoulder both trapezius muscles are mostly activated in pain patients. This co-contraction has to be reduced. The patient has to learn selective activation of the left or right trapezius muscles during separate elevation of the shoulder. Furthermore, patients exhibit prolonged activation of their muscles after a task [20]. In other words they are not able to relax their muscles, which they are not necessarily aware of because of the low level of activation. Nevertheless, according to Bults et al. low levels of prolonged activation contribute to chronic pain. However, in their study [20] they showed that patients with chronic neck and shoulder pain can alter their activity patterns if they receive feedback on their muscle relaxation times.

7.3 Game Design and Rehabilitation Key Concepts

As mentioned in Section 2.2 configurability, motivation, repetition, feedback and progress monitoring can be assumed key factors when designing serious games with a rehabilitation background.

We have derived certain design criteria from the above factors and requirements to be imposed on game play and elements interconnecting them with game aspects as described in the following subsections. Section 7.3.1 describes the importance of configurability and how it has been implemented in our game. A discussion of how we want to motivate the patient throughout the game can be found in Section 7.3.2. Then Section 7.3.3 describes how the concept of repetition is incorporated within our game, followed by details on the feedback provided to patients within the game in Section 7.3.4. We aim to provide feedback for the patient to ensure that he is capable to fulfill the exercise tasks correctly and to support motor learning effects (see also [57, 129]).

7.3.1 Configurability

Configurability and the right amount of challenge have been identified as important concepts in Section 2.9. Therefore, we made the game precisely configurable to the patient's capabilities. 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. For this purposes a scene has been implemented in the game as depicted in Figure 7.2, which is dedicated to measuring the physical capabilities of the patient with regard to the exercises. From the recorded baselines, levels of difficulty, parameters of the mini games and therapy goals can be calculated. The configuration employs a semi-automatic process in which the therapist chooses a parameter, which is followed by the avatar performing

the corresponding movement. The patient imitates the movement, while the system measures the parameter and the therapist confirms the correct execution. The therapy goal can be automatically calculated, manually entered or captured by the system, while the therapist physically supports the patient (e.g. lifts the arm of the patient). A fully automatic calculation based on measurements has shown to be not flexible enough. Firstly, it is rather difficult, even for an expert to give a general quantification of a goal. Secondly, the improvement of a patient's achievements is not continuous. Measurable differences are often very small and depend also to significant part on the patient's condition of the day or external influences (e.g. a smaller accident outside the game) of the patient. Therefore, we created options to correct the goals within the system. 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 can take a look at the data stored in the patient's profile to determine the progress and issues that have to be worked on. Providing the right amount of challenge is crucial for multiple aspects. Firstly, the physical challenge of the rehabilitation exercises should be close to capabilities to be effective, without however surpassing the pain threshold. Secondly, from the perspective of game design it is important to match the game play challenges precisely with the player's skills to achieve "flow", fun and avoid boredom and frustration as introduced in Section 2.9.

7.3.2 Motivation & Game Play

The core idea of applying serious games in this context is of course to provide motivation by making rehabilitation "fun" for the patients. This chapter largely discusses the game from a medical and technical perspective, however careful considerations in game design ensure that motivation is maintained throughout the game (as introduced in Section 2.2.1), e.g. through meaningful play and rewards (see Section 2.9 for a overview of design concepts for serious games). Facing and overcoming challenges is part of life and also an important part of games [29]. Crawford categorizes the dimension of challenges in games. In our game each mini-game provides it's own set of challenges for the player. Most of these challenges are of sensorimotor nature but also sequential reasoning is required in some cases. Conflict with an "active agent" (i.e. a human opponent or artificial intelligence) is also a central element of many games [29]. However, in our game we avoid conflict to leave the initiative with the user, which is important from a therapy perspective as stated earlier. Furthermore, adding an active agent would make the course of the game less predictable. However, for the evaluation of the therapy results within the user study it was important to have standardized reproducible situations. Nevertheless, in the future it would be interesting, if competition and other social effects could even improve motivation.

Considerations regarding game design are detailed in this and the following sections, where it seems appropriate.

7.3.2.1 Storyline & Game Concept

The overall concept of the game is that of an exploration game in an adventure setting. For the game environment an island scenario has been chosen, which is hypothesized to produce positive

Figure 7.2: Configuration scene used to adapt the game to the patients' capabilities.

associations of vacation, fun, relaxation and adventure. Furthermore, it is an environment largely decoupled from problems, which patients/players face in their every day lives. A tropical island is a pleasant, calm setting, and a place most people want to visit (warm and sunny, nature, peaceful). The main location was also chosen and designed graphically in order to be visually pleasing and calming. Patients may not be regular gamers and therefore will be less reluctant to enter the game world if it appears pleasing and welcoming. The fact that it is the player's choice (as mentioned in the introduction narrative) to sail to the island emphasizes that he is an empowered explorer rather than a stranded victim. Putting the patient in an active role is important from the rehabilitation perspective and hoped to transfer to the patient's real life.

In addition, an island scenario can be easily extended without much explanations in the story with additional islands or areas. Therefore, new exercises for the patients can be easily incorporated and combined in a modular fashion depending on the requirements of the patients.

The indication of chronic pain in the lower-back and neck is largely independent of age and gender. Therefore, it was one of the premises to create a storyline that appeals to a wider range of people. The combination of a realistic environment with mythological/fantastic elements offers a good option with many successful examples (Indiana Jones, Harry Potter, fairy tales, etc.). The storyline is inspired by the timeless Atlantis myth as well as Maya and Aztec cultures.

In the game the player arrives with his ship at a deserted island and discovers the remains of an ancient civilization. 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. When the player has collected enough items, he is rewarded by unlocking the next part of the story, which should provide additional motivation.

The basic idea of the implemented award system is to partially decouple the advance in the game from therapeutic progress, while still giving sufficient feedback on the performance through the game. Therefore, a patient who suffers a temporary decrease in his health condition is not additionally punished by losing in the game. This concept is also supported by Crawford [29] stating that "Losing should be a rare event, just frequent enough to maintain the illusion of risk, but not frequent enough to intimidate the player." Therefore, putting effort in the game is being rewarded even if the therapeutic progress lags a bit behind.

The advancement of the patient in the game is determined by two parameters: The number of artifacts, which have been collected in the mini games and the number of tables with the story that have been unlocked. These can be looked at anytime in the inventory on the ship and contain a narrative reward. Altogether there is nine tables, which tell the whole story of the Magupta Empire. Unlocking of the tablets is depending on the number and type of artifacts. A patient is also able to collect enough artifacts, if there is little or no therapeutic progress. However, as an incentive for the patient to execute the exercises precise and with full commitment collecting the items gets easier with a good exercise performance. On the other hand cleverness and tactics help with the advance in the mini games. Within the game there are various types of artifacts comprising different value and combining items of the same type yields bonus scores.

Figure 7.3) shows the game states and possible transitions between them. The game flow follows the same structural steps every time the patient plays:

1. Ship: It is a central element in the game, which also serves as a menu for navigation between the other states (Figure 7.4a). It links to calibration, status on progress (in-game achievements) and offers choices on the next task(s).

2. Island: For the game a 3D island world has been modeled in which the player can freely roam around (see Figure 7.4b). Embedded in this scenario are multiple mini-games, which can be activated by touching the corresponding symbols. Originally the patients were supposed to move around on the island by walking on the treadmill. However, this idea has been abandoned in an early design iteration, because this mode offers only limited control over the treatment protocol by the therapist. In the final prototype the player can be transferred to specific island locations using a graphical user interface and start the respective mini-games by using the in-game island map. The island itself can now be explored by the patient after each therapy session as an additional reward.

3. Mini-game world: After a mini-game is started, the player is transferred in the mini-game 3D world, much like the 3D island scene at the triggering location. The player or the practitioner may decide to end the mini-game at any point and return the island. When a mini-game is finished (successfully or unsuccessfully), the player may start again, exit back to the island, or be transported back to the ship.

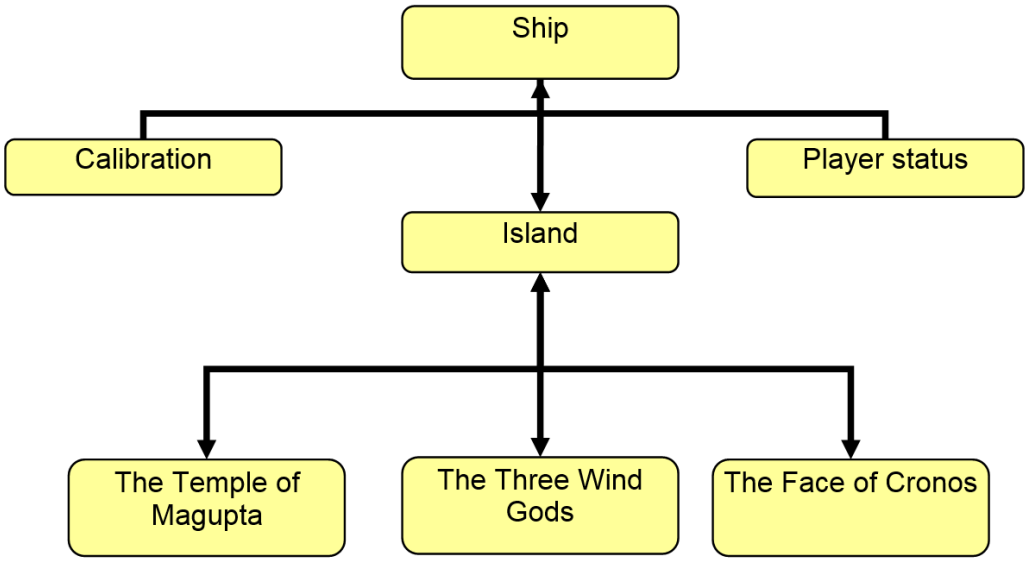


Figure 7.3: The different game states and possible transitions.

(a) The ship scene, that serves as central element in the game.

(b) 3D island world, that can be explored by the player.

Figure 7.4: Overview of game scenes.

4. Ship: The player is rewarded (if the task was successful or the effort was sufficient to merit a reward). The player can now choose to either do another task or end the game.

7.3.3 Repetition & Minigames

The mini-games embedded in the game environment contain the therapeutic core of the serious game. Every mini-game emphasizes one or more therapeutic goals. Exercises, that have been shown to be efficient during conventional therapy, have been integrated with the gameplay.

During the requirement analysis therapists have selected exercises for the mini-games according to the following criteria:

- Therapeutic benefit: Exercises should be efficient with regard to therapeutic progress
- Movement analysis: Biosignals and movement data is recorded in the patient profile. Thus, later analysis can provide insights on movement patterns and deviations with certain medical indications.
- General and specific applicability: Exercises should on the one hand cover a general part of therapy, but on the other hand contain also elements, which are specific to certain patient groups (e.g. Neck movements with whip lash patients)

During the mini-games the patients have to execute the exercises repeatedly as specified by the therapist in the calibration step to be successful in the game and ensure therapy progress. However, within the games it has been taken care that it is not just blind repetition of the same movements, but also a choice or cognitive component added. For example in the mini-game "Face of Cronos" the patient can select, which arm to use for the next move, which in turn influences the artifacts he/she can collect. In "Three Wind Gods" the patient has to memorize, which movements to perform next, thus providing a cognitive challenge and distracting him/her from the actual exercise.

The following describes the three mini-games for the chronic pain patients. The mini-games are not linked, in order to allow for various combinations and repetition of exercises.

- Temple of Magupta
- Three Wind Gods
- Face of Cronos

7.3.3.1 Temple of Magupta

The player explores a collapsing tunnel in an ancient temple (Figure 7.5) while controlling an avatar from third-person view. The path is split into five lanes. At any time, the player may command his avatar to change lane to the left or right, except if he is already in the leftmost or rightmost lane respectively. The goal for the player is to reach the room at the end of the tunnel within the given time. Along the path the player should pick up artifacts laying on the ground of the tunnel, while at the same time avoiding obstacles, such as holes in the floor and falling debris. Artifacts and obstacles may be found in any of the five invisible lanes. The player may have to change lanes frequently, in order to avoid obstacles in the current lane and to pick up approaching artifacts in other lanes and also plan ahead (e.g. focus on a certain type of artifact or combination thereof, which gain higher reward). The patient's voluntary walking velocity on a treadmill controls forward movement of the avatar. Lateral movement of the avatar is controlled by the patient's speech and not the movements for various reasons. Firstly, the walking pattern and activity of the trapezius muscle should not be influenced by additional gestures, because they are recorded for later analysis. Secondly, abrupt movements on the treadmill can be painful

Figure 7.5: Mini-game Temple of Magupta.

for the patient and threaten his safety. Safety and health of the patients has been one of the main premises throughout the development of the system and therefore we used a treadmill with safety rails on the sides. The main therapeutic goal of this mini-game is to improve the overall physical condition of the patient. He should be motivated to move more, increase walking velocity and thus achieve a training effect.

7.3.3.2 Face of Cronos

In this mini-game, the avatar is climbing a rock face as depicted in Figure 7.6. By using voice commands, the player chooses which arm to use (“Left” or “Right”). To help the avatar climb the rock wall, the patient has to reach overhead in the direction of the next virtual hand grip. Once that goal is accomplished, the avatar climbs one step higher on the face. The distance to the handhold as well as the height of the rock face depends on the configured parameters. An artifact is earned as a reward for every successful overhead reach. Due to the number of repetitions set by the therapist during calibration, the patient will have a certain number of reaches he can use to reach the top of the cliff. To be able to arrive at the top within the limited number of reaches the player has to achieve a certain height every time. If a reach goes below baseline,

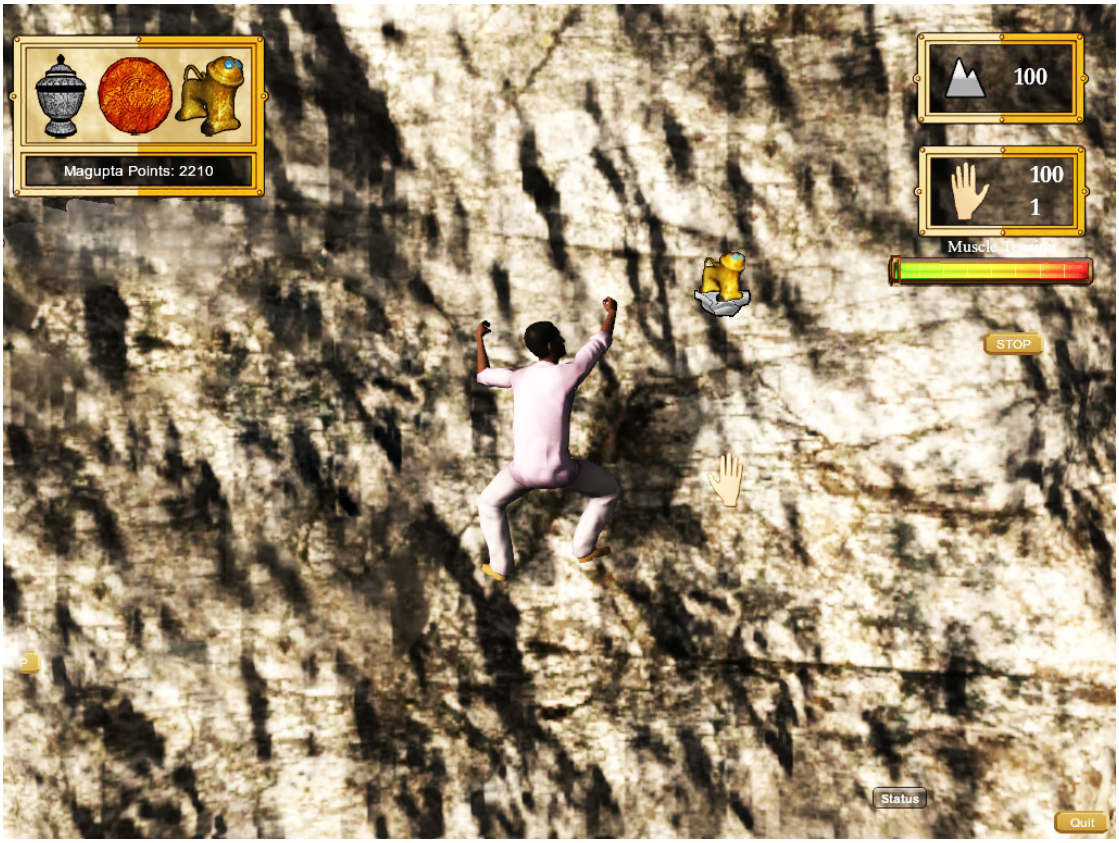


Figure 7.6: Mini-game Face of Cronos.

the following attempts will have to compensate. This might be easy at first, but the player will become tired, which will gradually increase the difficulty. The same applies for the number of repetitions entered during calibration by the therapist. The game mechanism challenges the patient to perform the movements correctly until the very last. Reaching the top will award the player with bonus artifacts, which contribute to the overall game progress and will eventually unlock tablets. In case a player does not make it to the top of the cliff he is not rewarded with bonus points, but still keeps the artifacts collected during the climb. Additionally, the patient receives real-time feedback about the muscle tension in both trapezius muscles. The patient is instructed to relax as quickly and as much as possible in between movements, in order to gain consciousness about his or her muscle tension. This is especially important after a larger number of reaches, when relaxation intervals become increasingly important for sustaining the full movement range. The underlying goal of this mini-game is to improve overhead reaching ability.

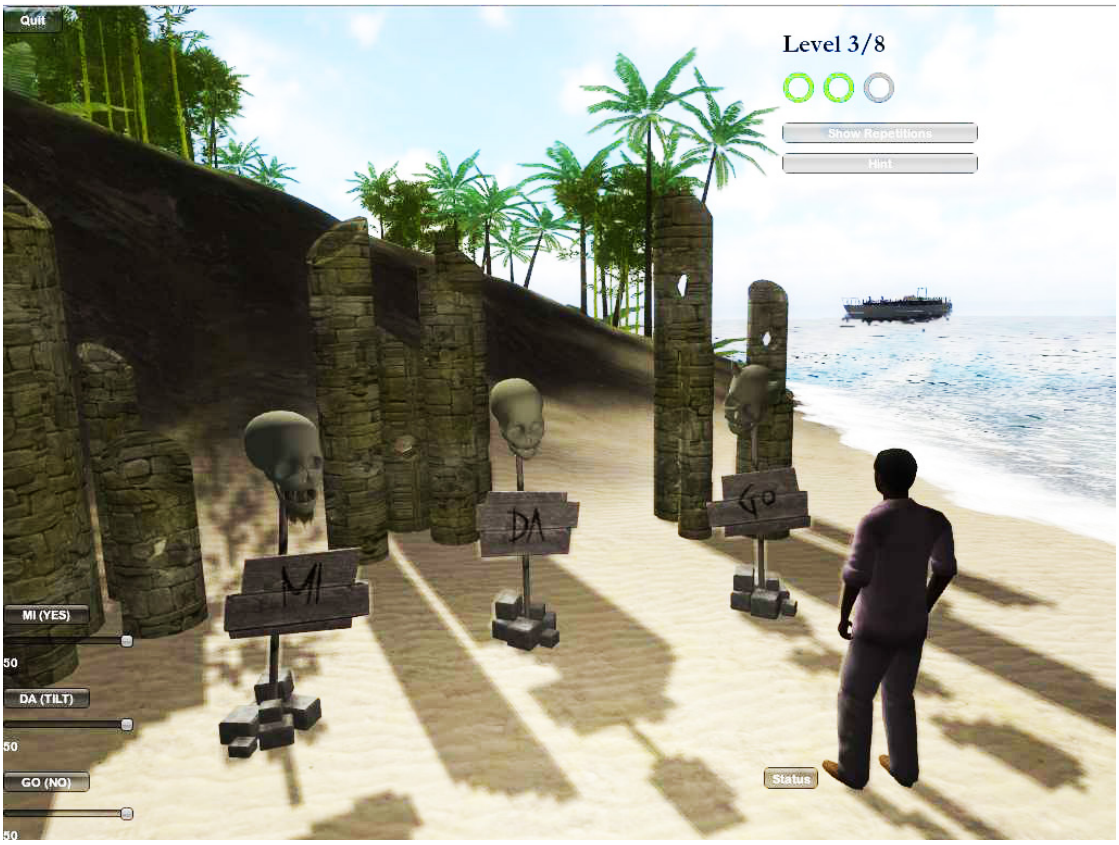


Figure 7.7: Mini-game Three Wind Gods.

7.3.3.3 Three Wind Gods

This mini-game's main game element is a stone structure with wind pipes located at the beach (Figure 7.7), which can be activated by the player through different head movements. Each pipe produces a unique sound and the challenge is to play them in a certain sequence or melody. First the game presents the sequence visually and through audio and complexity increases with the level. The therapist adapts the number of levels and therefore repetitions depending on the abilities and therapy goals of the patient. The patient has to carry the movement through the whole configured range (e.g. "Yes"-movement over 95 degrees). Three characters show the head movements which the patient has to reproduce. These movements, corresponding to the characters, are flexion–extension, rotation, and lateral flexion–extension of the neck. For every successful reproduction of a movement, an artifact is earned as a reward. Furthermore, the game rewards smooth movement with bonus artifacts. The overall goal of this mini-game is to improve the patient's neck mobility.

7.3.4 Feedback & Game Output

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. In most cases transformed feedback, as introduced in Section 2.6, is displayed through game objects. Feedback on results is also important within the therapy process as explained in Section 2.2. Therefore, the patient may view his progress in the game or have a look at the profile after each game session. The profile presents objective data, recorded during gaming (e.g. reaching ability, cervical range of movement. Thus feedback on the patient's therapy progress provides knowledge of results. In addition a number of game elements and mechanisms provide feedback to the patients during the mini-games producing knowledge of performance.

Feedback is also important from a game design perspective. Salen et al. emphasize the concept of meaningful play for successful game design [119], as introduced in Section 2.9. They state that "Meaningful play in a game emerges from the relationship between player action and system outcome". They point out the importance of discernibility of a player's actions for meaningful play. In other words if a user can't see his actions reflected in the games feedback, interaction with the game becomes random and most likely frustrating for the user.

Therefore, in our game objects, animations, progress bars, scales, sounds give various forms of feedback to the player. In addition, the player's progression within the mini-games is integrated with the larger context of the game, with integration being the second key factor of meaningful play. Scores collected in individual mini-games reveal new parts of the story after the mini-game is completed.

The following paragraphs present details on the feedback provided in every mini-game.

7.3.4.1 Temple of Magupta

The mini-game provides visual feedback by multiple objects and interface elements. The animation of the avatar gives a rough estimate of the current speed, which should be sufficient for more experienced players. Additionally, there are two speed indicators in the game. One for the avatar (the character in the game), and one for the player (the speed the patient walks with on the treadmill). The player's speed indicator has three levels: red, yellow and green. The red speed level covers velocities from zero to 10% below baseline speed which the patient is walking on the treadmill. As long as the player's speed is in the red level, the avatar's speed will decrease. The yellow speed interval is from 10% below baseline to 10% above baseline. As long as the player keeps within this interval, the avatar speed will not change. Finally, the green speed level ranges from 10% above baseline speed and above. As long as the player keeps within this interval, the avatar speed will increase until it reaches maximum. In addition, there is a time-limit for the player to explore the temple, which is displayed in minutes and seconds. Furthermore, graphic elements show how many artifacts of each type have been collected and give an estimate how much are needed for a bonus.

7.3.4.2 Face of Cronos

The game displays hand position and muscle activity through graphic elements. Animations show the avatar climbing, after a new hold has been reached. Additional feedback such as the

remaining distance and number of reaches is provided in textual form.

7.3.4.3 Three Wind Gods

Game objects and a graphic scale show the range of neck movement, which the patient has already covered. In addition to visual feedback, audio feedback is added, because due to the head movements the display is not always in view. To indicate when the player has reached the required ROM for a movement, a brief 'scoring' sound is played, which at the same time is part of the melody. At the start of the level the player will hear the sounds that he needs to reproduce.

7.4 Evaluation in a Preliminary Medical Study with an Early Game Prototype

7.4.1 Evaluation Design:

Preliminary tests have been conducted to assess the MoCap system, the workflow and an early version of the game prototype. While some quantitative measures have been taken, we have concentrated on evaluating the usability using qualitative measurements. The evaluation was conducted with experts in movement science from Roessingh Research and Development (RRD). In total three male pain subjects (age 23-60) and seven healthy participants male and female (age 22-30) volunteered to participate in the preliminary tests. The three pain patients, had pain complaints regarding neck, shoulder or back at least one week during the month prior to the evaluation (median 20 days (7-20)). At RRD premises the motion capture system and the game environment was installed (Figure 7.1). With this setup the patient could test the game, after being instructed and while being monitored by a therapist. For the evaluation an iotracker system and TMSI MOBI as described in Section 6.6.4 were used. A projector displayed the game content to the patient, while the therapist had her own control monitor. A treadmill was used for one of the games.

The protocol for conducting the preliminary evaluation of the game followed the basic workflow described in Section 4.1. After having finished the three mini games the subject filled in a self composed questionnaire. This questionnaire consists of three parts: questions (1) specific for the different elements of the game/system e.g. suit, EMG, treadmill, (2) about the emotional perception of end users with gaming and (3) about preference for therapeutic implication(s).

Items were rated on a seven point Likert scale from one (strongly disagree) to seven (strongly agree), on a visual analogue scale (VAS) or on yes/no.

7.4.2 Results

The overall median duration of the MoCap calibration was 4:25 minutes. On average three calibrations were performed and 87.5% of all calibrations were correct. All participants were capable to perform the calibration exercise and qualified this movement as not difficult.

During the evaluation the update frequency of the MoCap system was stable at 50Hz. No major tracking errors have been detected, the system was running robustly. Visual inspection of

the motion data showed smooth movements with little jitter and evaluation of the joint angles showed also little jitter and high relative accuracy.

The duration of the mini games “Temple of Magupta”, “Face of Chronos” and “Three Winds Gods” were respectively; 2:30 minutes, 2:30 minutes and 4:05 minutes.

Overall the participants were modest regarding the enjoyment they perceived while interacting with the games. The overall median level of enjoyment for the three mini games was between a score of 5.7 and 6.05 on a 10 point VAS scale. However, it has to be stated at this point, that neither the graphical models were finished at the time of the evaluation, nor the reward system and levels of difficulty. The mini game “Temple of Magupta” was rated the most enjoyable mini game by 50% of the participants. It was the most active and dynamic out of the three mini games. Following verbal and written feedback by the participants game objects, which were too small or poor in contrast have been adjusted accordingly. Patients rated the “Three Wind Gods” to take the most effort and to be most exertive out of the three mini games. In other words, this game was labeled to be (too) challenging to play (it has also a cognitive component) and physically demanding. They had to approach their maximum range of motion, which can be painful especially for the patients (score of 7.1 on effort expectancy and 6.3 on perceived exertion). According to the therapists, this might be explained by the fact that the game puts much emphasis on the painful movement (i.e. range of motion). Therefore, the mini-game has been made a little bit easier and the required ROM reduced by a few percent. Another explanation for the lack of enjoyment is, that the movement sequences were too long and as Crawford points out in [29]: “Most people find highly sequential tasks, such as long calculations or memorizing complex sequences of actions, to be a tedious challenge.” Therefore, additional visualizations help the patient to reproduce the correct sequence in the final prototype. Patients rated the “Three wind gods” to be of lowest therapeutic relevance. One explanation for this is the fact that one of the “Three wind gods” was hardly visible on a projection screen, causing subjects to make mistakes more easily. Therefore, for the final game a visual/audio guidance for the patient has been implemented and game elements, that were considered to small or low in contrast, have been improved/exchanged. Results presented in the next section indicate that this adaptation of the challenge has made the mini-game more enjoyable.

The majority of participants (90%) would recommend this “serious gaming modality” to be added to their current rehabilitation treatment. One participant stated that a “serious gaming modality” would motivate them to increase their training intensity. About half of the patients would recommend a “serious gaming modality” to a friend (66.7%) or to another patient (50%). Most participants believed it is very valuable to see movement (which they execute) to be reflected in the game.

Feedback collected in this early user study has been of very high importance for development of the final game prototype and various improvements have been implemented to improve the user experience.

7.5 Evaluation in a Final Medical Study with the Final Game Prototype

The evaluation was conducted by experts in movement science from RRD with Stephanie Jansen-Kosterink being the principal investigator. This study is believed to be the first evaluation of a serious game for patients suffering from chronic pain and comprises a stage 1–2 evaluation, according to the staged approach of tele-medicine evaluation proposed by DeChant et al. [33]. The primary aim of this pilot study was to explore the user experience in terms of usability, satisfaction, level of motivation, and game experience of patients playing the game. The secondary aim of this pilot study was to explore the progression in terms of the performed motor skills (walking velocity, overhead reach ability, and cervical range of motion (CROM) and the clinical changes (physical condition, level of disability, and pain intensity) brought about in chronic musculoskeletal pain patients playing the game over 4 successive weeks. The findings of this study have been published in [131], [128] and [63].

7.5.1 Study Design

7.5.1.1 Participants

Patients were recruited from a local physical therapy practice and by an advertisement in the newsletter of a patient association for chronic pain patients. Interested patients could contact the researchers and received an information letter concerning the study. Only those patients 18 years and older and with low back pain or pain in the neck/shoulder region for at least 12 weeks (without specific pathological causes) were included. Patients were excluded if they (1) had an insufficient understanding of the Dutch language, (2) had visible impairment that inhibits the perception of the screen on which the game is projected, or (3) were receiving (physical) therapy for their physical complaint at the time of the study. Because of the explorative character of this study, no sample size calculation has been conducted beforehand. To be able to answer the objectives of this study, the goal was to include at least 10 patients. The medical ethical committee approved the study. All patients gave their informed consent prior to participation.

7.5.1.2 Setup

At RRD premises the motion capture system, biosignal acquisition device and the game environment as described in Chapters 4 and 6 had been installed. In this environment the patients could play the game, after being instructed and while being monitored by a therapist. A video projector was used to display the game content to the patient, while the therapist had her own control monitor. Figure 7.1 shows this setup. For the mini game “Temple of Magupta” the patient had to walk on a treadmill.

Relevant motions of the patient’s body and muscle activation levels control the game. To track the patients’ movements while playing they were equipped with a tight-fitting suit (a jacket, a pair of trousers, and a cap) with 36 reflective markers attached to it as shown in Figure 7.8. Easy-to-put-on suits were available for male and female patients and in various sizes (small, medium, large, and extra large). The suits were washable. At the start of the treatment a clean

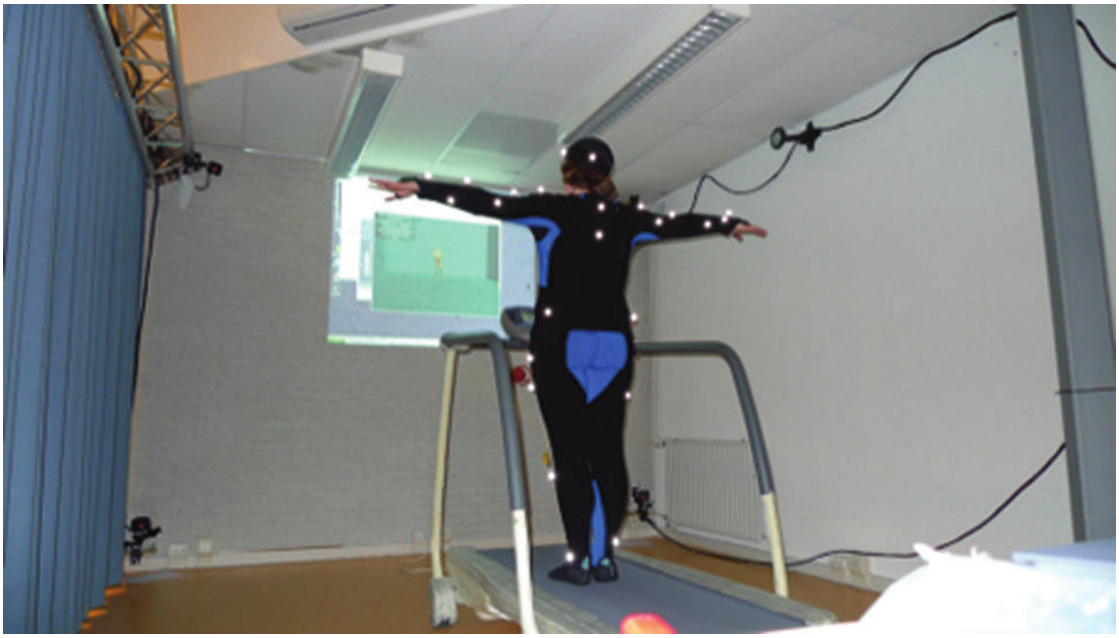


Figure 7.8: Setup with the MoCap system and a user wearing the MoCap suit.

suit was provided to every patient; on the patient's request the suit was washed during the treatment period. Furthermore, surface electromyography electrodes are placed on both left and right upper trapezius muscles, on the halfway point of the line between the spinous process of C7 and the acromion. A reference electrode is placed over the spinous process of C7 and the signal is sent to a computer wirelessly.

7.5.1.3 Treatment Protocol

The patients were asked to visit Roessingh Research and Development lab (Enschede, The Netherlands) and to play the game for four weeks with a frequency of one or two times a week. After 4 weeks of training with this frequency, first clinical changes can be expected. In addition, this amount of sessions should enable patients to give a reliable judgment concerning their experience. During every game session, a therapist assisted the patient. The first gaming session was focused on the calibration of the MoCap system and to assist the patient become acquainted with the game. Subsequently, by using the baseline and goal-setting module, the individual baseline values were assessed, and individual goals were configured for each motor skill. This module also automatically updates the baseline values of the patient if the patient performs at beyond the baseline level. Then the patient was introduced to the mini-games, and the aim of every mini-game was explained. During subsequent sessions, the patients played every mini-game at least three times. Depending on the primary complaint (low back or neck/shoulder pain) or patient's preference, a mini-game might be played more than three times. Each gaming session lasted between 45 and 60 minutes (15-20 minutes preparation / 30-40 minutes gaming).

7.5.1.4 Measurements

User experience ISO 9241-210 defines user experience as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service." In the assessment of the user experience during the game we focused on usability, satisfaction, level of motivation, and gaming experience. The usability of the game was assessed against the System Usability Scale (SUS) [19]. The SUS presented 10 statements about the perceived usability of the game. Patients could indicate on a scale of 0–4 to what extent the presented statements were true for them. To obtain the final SUS score, the sum of the patient's answers was multiplied by 2.5. The SUS score ranges from 0 to 100 (low and high usability, respectively). The English version of the SUS was translated into Dutch, as there was no validated Dutch version available. The overall satisfaction with the game was assessed by a request to rate the game on a scale from 0 to 10 (low and high usability, respectively) and an open-ended question asking about the overall experience of the game. The level of motivation was assessed by posing two questions. For the first question, patients rated their level of motivation related to the game on a 7-point Likert scale, ranging from "demotivating" to "motivating". The second question was answered with yes or no: "Did the game motivate you to perform your exercises?" The game experience of the patients during gaming was assessed by the enjoyment, frustration, environment (graphics and sounds), and game play (scenario and rules) scale of the Core Elements of Gaming Experience Questionnaire (CEGEQ) [25]. This questionnaire presents 17 statements. Patients could indicate on a scale ranging from 0 to 7 to what extent these statements were true for them. The summed score per category gave a view of each patient's overall gaming experience. The English version of the CEGEQ was translated into Dutch, as there was no validated Dutch version available. All the questionnaires were assessed immediately after weeks of gaming (post-test).

Game output The progression on the performed motor skills was assessed by analyzing the game data. After every game session, the game data were saved in a patient-specific folder. In this folder were saved the walking velocity ("Temple of Magupta"), the movement time and velocity and three-dimensional wrist position ("Face of Cronos"), and the rotation of the head around the three axes ("Three Wind Gods"), as well as other game data.

Clinical changes The physical condition of the patient was assessed by the 6-minute walk test [8]. The objective of this test is to walk as far as possible for 6 minutes on a flat surface such as a hallway. During the test, patients were permitted to slow down, to stop, and to rest if necessary.

The subjectively experienced disability of patients with pain was assessed by a generic disability questionnaire, namely, the Pain Disability Index (PDI) [145]. The PDI is a selfrating scale. Answers are provided on a categorized 11-point scale with "not disabled" and "fully disabled" at the extremes. In a chronic pain population, the psychometric properties of the PDI appeared to be sufficient [145]. Pain intensity of the back region or neck/shoulder region was assessed by means of a visual analogue scale (VAS) [46]. Patients were asked to rate their experienced level of pain during the last month. The VAS consists of a 10-cm horizontal line with "no pain" on the left and "worst pain ever" on the right extremity of the line. Psychometric properties have proven to be sufficient. The 6-minute walk test, PDI, and VAS were assessed prior to (pre-test) and immediately after 4 weeks of gaming (post-test).

Analysis To assess the user experience, the mean scores post-test of the assessed questionnaires were calculated. Because of the pilot characteristics of this study, all patients were asked to play all three mini-games, even if the motor skill(s) requested in the mini-games did not correspond with their specific needs. For this reason the progression on the performed motor skills of the game are presented separately for the whole group and for the patients with a significant impaired function for these motor skills.

For the "Temple of Magupta" mini-game, the average walking speeds for each game week are presented. The average walking speed was calculated over the total "Temple of Magupta" mini-game, per session.

For the "Face of Cronos" mini-game, the average overhead reaching heights, based on wrist positions, for each game week are presented. The wrist position was determined as the maximum wrist height per movement relative to the shoulder. The maximum height of every reach was defined as the point, between the start and end of a movement, in which the velocity of the wrist was 0 m/second. For every game session, an average value was calculated to describe the reaching heights.

For the "Three Wind Gods" mini-game, the average range of motion (in degrees) of the cervical axial rotation from left to right (no movement) for each game week is presented. For every game session, an average value was calculated to describe the CROM. Data processing and calculation were done using Matlab software (version R2008b; MathWorks, Natick, MA). To investigate the changes of game output over the weeks, mixed-model analysis for repeated measures was used. Time of measurement (week) was used as a within-subjects factor. Post hoc comparisons were made when required, and Sidak's adjustments were used to correct for multiple tests. At a group level, the overall clinical effect of our game over time (pre-test versus post-test) on physical condition, disability, and pain intensity was analyzed using a paired non-parametric test (Wilcoxon). SPSS version 19.0 software (SPSS, Inc., Chicago, IL) was used for statistical testing. Alpha was set at 0.05 to test for statistical significance.

7.5.2 Results

Ten patients (two male and eight female) participated in this study. All patients met the predefined inclusion criteria and completed the 4 weeks of gaming. The mean age was 54.9 years (standard deviation [SD] 11.8; range, 27–68). Six of the patients reported primary neck/shoulder complaints, and four patients reported primary low back complaints. The average pain intensity of the patients was 5.9 (SD 2.1) and the average disability score was 27.3 (SD 13.5), indicating mild to moderate disability levels. All patients were able to walk without a walking aid. Four of the patients were employed and worked for 28–40 hours per week, three of the patients were retired, and three patients were unemployed (Table 7.2).

7.5.2.1 User Experience

With respect to the reliability analysis of the SUS, Cronbach's alpha of the 10 items was 0.5, indicating poor reliability. The usability of the game was rated good (SUS score $\geq$71.4 [10]), with a mean SUS score of 77.5 (SD 9.5; range, 60.0–97.5). The overall satisfaction with the game was high. The patients gave the game an average rating of 7.6 (SD 0.7; range, 6–8) out of

Table 7.2: Demographic Characteristics of the Included Patients.

Characteristic	Value
Age (years)	54.9 (SD 11.8; range, 27–68)
Gender	20% male
	80% female
Complaints	60% neck/shoulder pain
	40% low back pain
Pain intensity	5.9 (SD 2.1; range, 1.6–8.0)
Disability score	27.2 (13.5; range, 6–55)
Work status	40% employed
	30% retired
	30% unemployed

10. Four patients responded to the open question about the overall experience of the game. The patients stated:

"The game distracted me, I was less aware of the pain and therefore I was able to relax and play the game." (Patient number 3)

"I liked to play the game." (Patient number 4)

"The Temple of the Mapugta mini-game I enjoyed most, the other two games I enjoyed less. These mini-games were insufficiently stimulating for me." (Patient number 5)

"Playing the game caused distraction from my pain, therefore I was able to execute the requested exercises better; I walked longer and stretched my neck more." (Patient number 9).

Nine of the 10 patients found the game motivating; one patient gave a neutral answer. All the patients found that the game motivated them to perform the requested motor skills. The outcomes on game experience are presented in Figure 7.9. With respect to the reliability analysis of the CEGEQ, Cronbach's alpha of the 17 items was 0.9, indicating good reliability. Patients enjoyed playing the game and experienced very little frustration. The high scores indicate that the patients were satisfied with the game environment (graphics and sounds) and game play (scenario and rules).

7.5.2.2 Progression on Performed Motor Skills

The results on the progression in performed motor skills (walking velocity, reaching height, and CROM) are presented first for the whole group and second for those patients with relevant impairment.

7.5.2.3 Walking Velocity

Group Mixed-model analysis for repeated measures showed that the scores on walking velocity did not change significant over time ($P \geq 0.22$). The average baseline (voluntary) walking velocity was 3.4 km/hour (range, 2.5–4.5). In the first week, the average walking velocity during gaming was 4.0 km/hour (SD 0.6; range 3.2–5.2), and in the final week the average walking velocity during gaming increased to 4.8 km/hour (SD 0.6; range, 3.7–5.6) (Figure 7.10 a).

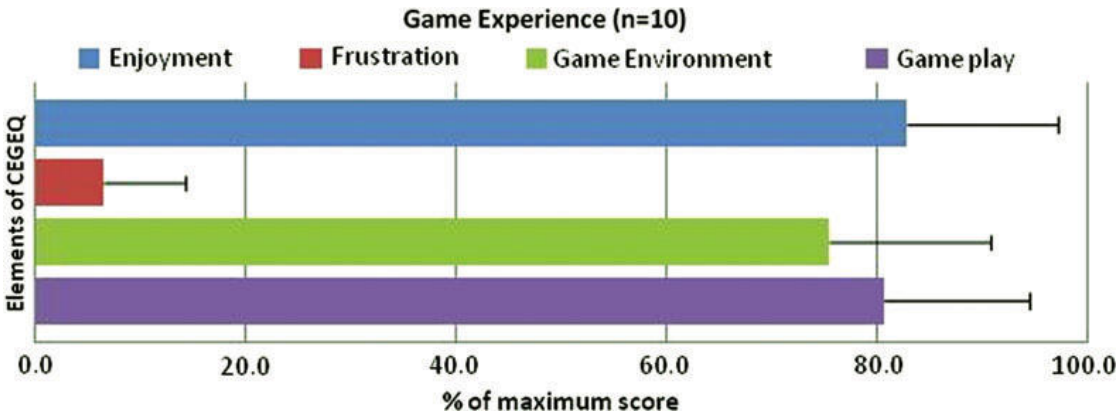


Figure 7.9: Overview of overall scores on the Core Elements of Gaming Experience Questionnaire (CEGEQ) subscales of enjoyment, frustration, game environment, and game play.

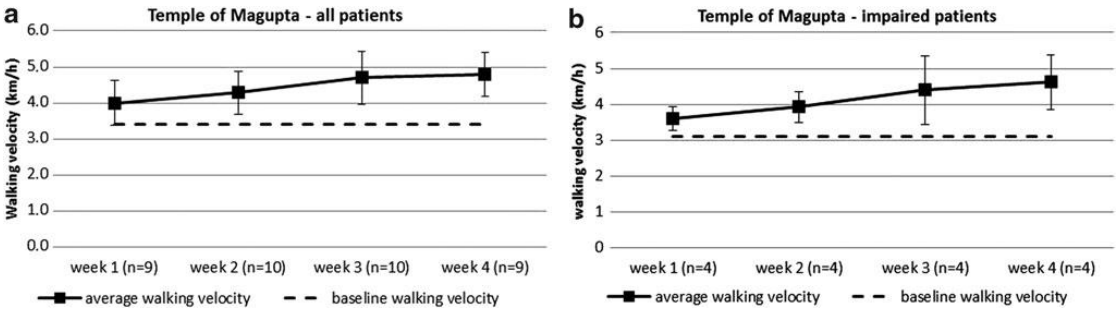


Figure 7.10: Average walking velocity of (a) all patients and (b) the patients with an impaired walking velocity during the "Temple of Magupta" mini-game.

Impaired patients Four patients (numbers 2–4 and 10) had an impaired walking velocity of 4.2 km/hour or less on the 6-minute walk test performed pre-test [67]. Looking at their walking velocity during playing the game, game data showed an average baseline walking velocity of 3.1 km/hour (range 2.5–3.5). In the first and final weeks these values were 3.6 km/hour (SD 0.3; range 3.2–4.0) and 4.6 km/hour (SD 0.8; range, 3.7–5.3), respectively (Figure 7.10 b).

7.5.2.4 Reaching Heights

Group Mixed-model analysis for repeated measures showed that the scores on reaching height did not change significant over time ($P \geq 0.17$). The average baseline reaching heights for the left and right arms were 2.05 m (range 1.91– 2.16) and 2.05 m (range 1.93–2.14), respectively. For the left arm, the average reaching height in the first week of gaming was 2.08 m (SD 0.03; range 2.02–2.13). In the final week of evaluation, the average reaching height of the left arm increased to 2.10 m (SD 0.02; range 1.98–2.15). For the right arm, the average reaching height in the first week was 2.05 m (SD 0.02; range 1.85–2.15). In the final game week, the average reaching height of the left arm increased to 2.10 m (SD 0.01; range 2.03–2.17) (Figure 7.11a).

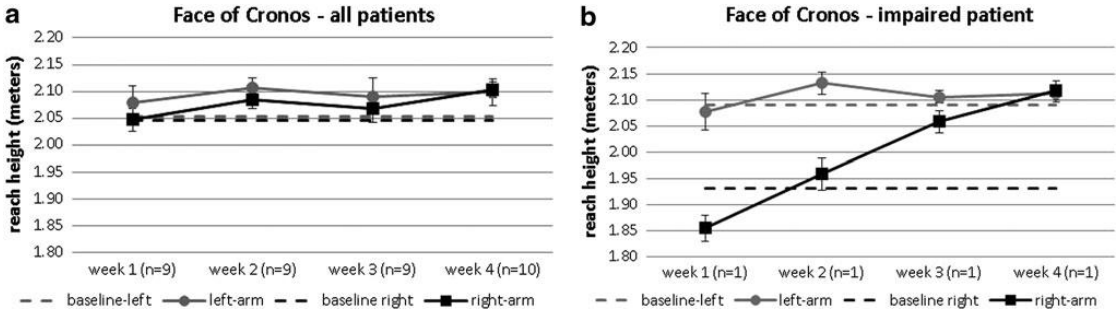


Figure 7.11: Reaching height overhead of (a) all patients and (b) the patient with an impaired reaching height during the "Face of Cronos" mini-game.

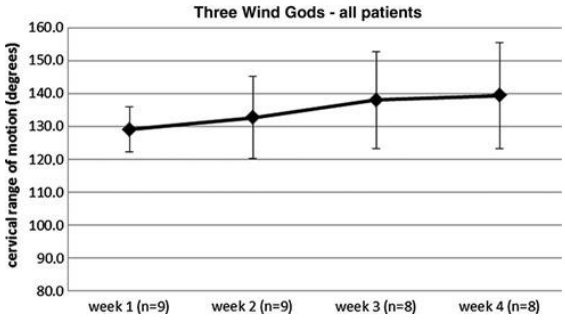


Figure 7.12: Cervical range of motion of all patients (all had impaired motion) during the "Three Wind Gods" mini-game.

Impaired patient Only one patient (number 2) had a discrepancy of overhead reaching height between the left and right arm in the first week. The reaching height of the right arm was 23 cm below the measurement of the left arm. The reaching height of the right arm of this patient in the first week was 1.85 m, and increased to 2.12 m in the final week (Figure 7.11b).

7.5.2.5 CROM

Group/impaired patients Mixed-model analysis for repeated measures showed that the CROM scores changed significantly over time ($P \leq 0.03$). During the first week, the CROM for cervical axial rotation, from left to right, during gaming was 129.1° (SD 6.8; range, 119.9°–138.5°). The CROM for cervical axial rotation, from left to right, of a healthy patient is 151.7° [137]; all patients had an impaired CROM ($\leq 151.7^\circ$). During the final gaming week, the CROM increased to 139.5° (SD 16.1; range, 121.0°–161.2°) (Figure 7.12). During the final week, three patients (numbers 2, 4, and 9) reached the CROM cutoff point ($> 151.7^\circ$) for healthy patients [137].

7.5.2.6 Clinical Effectiveness

The physical condition of the patients was assessed by using the 6-minute walk test. Pre-test, the average walking distance was 445 m (SD 77). At post-test, the average walking distance

Table 7.3: Outcome on Pain Intensity, Pain Disability, and 6-Minute Walk Test.

	Mean (SD)	
	Pre-test	**Post-test**
Pain intensity	59 (21)	50 (24)
Pain disability index	27.2 (13.5)	26.3 (15.0)
6-minute walk test	445 (77)	465 (46)

increased by 20 m to 465 m (SD 46). However, this difference was not significant (P = 0.212). The PDI decreased by almost 1 point at post-test. Nevertheless, the difference between pre-test and post-test scores for disability was not significant (P = 0.505). After 4 weeks of gaming, the perceived pain intensity of the patients decreased from 59 (SD 21) on a 100-mm VAS to 50 (SD 24) as shown in Table 7.3. Again, this average difference was not significant (P = 0.284).

7.5.3 Discussion

The presented pilot study focused on a first evaluation of the game for patients suffering from chronic musculoskeletal pain. The primary aim was to explore the user experience of the patients with the game. The secondary aim of this pilot study was to explore the progression in terms of the performed motor skills (walking velocity, overhead reaching ability, and CROM) and the clinical changes (physical condition, disability, and pain intensity) induced in chronic pain patients using the game for 4 weeks. Patients experienced the game as positive. They rated the usability of the game as good and the games clearly motivated patients to perform their exercises. Furthermore, patients enjoyed playing the game and liked the game environment and game play.

Despite the short training period, overall the patients made a nonsignificant progression in terms of the requested motor skills in the mini-games during the 4 weeks of gaming, especially those patients with impaired motor skills. After 4 weeks of gaming, generally patients were capable of walking faster and reaching higher and experienced an increase in neck mobility. Based on our results, it is expected that serious gaming has a true potential for physical rehabilitation This is especially true when used in combination with telerehabilitation applications that enable home-exercising. Home-based exercise programs are known to be effective [84], but the overall low conformance with such programs remains problematic [142]. Furthermore, a high conformance to a rehabilitation program has a positive effect on clinical outcomes [28]. Serious games in rehabilitation, such as ours, encourage patients to perform their exercises, so they have the potential to overcome the generally low compliance with home-based exercise programs. Another positive aspect of our serious game is the availability of game data. These data provide the therapist with detailed information on the progression of a patient in terms of the various trained motor skills. By using the available game data, the therapist can better align the game session to the needs of the individual patient, and the transparency of the treatment is increased, which matches the current trend in healthcare. However, none of the commercially available exergames currently provides the healthcare professional with this type of game output.

Previous randomized controlled trials have shown the potential of games targeting the rehabilitation of stroke patients [75, 121] or patients with acquired brain injury [48]. Clinical

trials to test the benefits of serious games in rehabilitation are necessary before such games are incorporated into rehabilitation programs [88].

Our pilot study is a first step toward the implementation of serious games in the physical rehabilitation of chronic musculoskeletal pain patients. The framework for telemedicine evaluation proposed by DeChant et al. [33] was used, and in line with this framework, the presented pilot study was a stage 1–2 evaluation. In the terminology of DeChant et al., [33] the evaluation of an application starts with an evaluation of the technical efficacy (accuracy and reliability) of the application and an evaluation of the primary objective of the application in terms of access, quality, or cost (stage 1–2). During the subsequent deployment, a comprehensive evaluation is necessary, using multiple end points such as the quality, accessibility, and costs of this healthcare approach (stage 3). The final step in the evaluation of an application is to examine whether the overall evaluation of an application in one system can also apply in other settings (stage 4) [33]. Even though the framework is designed for telemedicine evaluation, it can be adopted for evaluation of serious games in healthcare and also help other researchers to organize the evaluation of their serious games for rehabilitation in a clinical setting.

Although the sample size of our pilot study is low and there was no control group, it still extends the knowledge about the use of games in the physical rehabilitation of chronic pain patients [5]. By playing our game, patients made a progression in the requested motor skills. In rehabilitation a progression of 15 percent is often considered as clinically relevant [164]. Given our results on a group level, the progression made on walking velocity is clinically relevant. On an individual level 70 percent of the patients made a clinically relevant progression on walking velocity, 10 percent of the patients made a clinically relevant progression on reaching height, and 30 percent of the patients made a clinically relevant progression on CROM.

One of the limitations of this study is the use of nonvalidated Dutch versions of the SUS and CEGEQ, because no other validated Dutch questionnaires to assess usability and game experience were available. Given the Cronbach's alpha of these questionnaires, the reliability of the SUS and CEGEQ were indicated as poor and good, respectively. Therefore the use of this Dutch version of the CEGEQ is recommendable, but for the future, use of the current Dutch version of the SUS should be reconsidered.

A next step in the evaluation of our serious game is to compare it with conventional physiotherapy for patients suffering from chronic musculoskeletal pain and to assess them on clinical benefit, user experience, and costs (stage 3). The following adjustments could increase the potential of our game as a tool for rehabilitation. First, the current version of the game only involves three mini-games, and an increase of the number of mini-games would be beneficial. With more mini-games available, a game session could be better adjusted to the rehabilitation goals of the individual patient, and that patient's treatment protocol can be refined. Second, the duration of the treatment protocol in this study was 4 weeks. Patients visited the Roessingh Research and Development lab to play the game one or two times a week over 4 weeks. Because of the positive effects of intensity, frequency, and duration of training on physical fitness [77], it can be assumed that extending the treatment protocol (for example, duration of 6 weeks instead of 4 weeks and frequency of at least twice a week) could further positively influence the outcome.

A final suggestion is to adjust the game for remote physical rehabilitation. In the current setting, patients were dependent on the availability of the therapist and the Roessingh Research

and Development lab to play the game. Besides, an elaborate MoCap system was needed to generate input for the game. For a remote physical rehabilitation setting, there must be an easy-to-use and cheap alternative for this MoCap system, such as Microsoft's Kinect. In a previous study, Microsoft Kinect was integrated into our game and tested as an alternative low-cost MoCap system. In this setting, two of the mini-games ("Temple of Magupta" and "Face of Cronos") could be controlled by the requested motor skills [131]. Concerning our game as a tool for remote physical rehabilitation, a next step is to evaluate the game in combination with low-cost maintenance-free motion capturing systems in a home setting and see if the outcome on accessibility and quality are comparable with the outcomes of this study. Our existing knowledge about telerehabilitation suggests that our game has the potential to increase the quality and accessibility of health-care and perhaps lower costs. Patients can play the game during a self-scheduled time span in their own environment. This would fit with the current trend of self-management of the patient [68].

In conclusion, our serious game for chronic pain rehabilitation has demonstrated potential efficacy in a small sample of adults with musculoskeletal pain. It could be a novel tool for improving outcome of physical rehabilitation, because it motivates patients to perform their exercises, and as a result, their motor skills and physical condition improve.

CHAPTER 8

Conclusion

8.1 Summary and Discussion

This thesis has focused on the development of methods allowing full body interaction emphasizing algorithms and implementation of Motion Capture (MoCap) systems. A framework has been presented, that facilitates the implementation of (serious) games and applications using MoCap and other Virtual Reality (VR) technologies. Furthermore, design, development and evaluation of a game for chronic pain rehabilitation based on our framework was described.

Key concepts of VR technology in rehabilitation, such as motivation, repetition and feedback and other design considerations of serious games were identified, discussed and their application exemplified.

The development of an affordable full body MoCap system, based on optical tracking data, was described including a detailed description of the algorithms and methods for skeleton calibration and tracking. In that, we attempted to bridge the gap to commercial MoCap systems, for which little has been published on the applied algorithms. Furthermore, our MoCap system was designed and developed with a focus towards the application in a serious game for rehabilitation. Therefore, features such as a flexible marker-setup and a workflow that can be easily handled in an every-day clinical setting have been emphasized. The system worked robustly over an extended period of time during evaluation with a serious game targeting patients suffering from chronic musculoskeletal pain and showed good usability.

Home-based exercises were introduced as an important and interesting application area of serious games for rehabilitation, that can help guide a patient in his relearning process (e.g. correcting errors in movement patterns), which is done by a therapist during conventional occupational or physical therapy. Marker-based infrared optical MoCap systems, however, are too expensive and difficult to maintain in a home environment and we therefore evaluated Microsoft Kinect as an alternative MoCap solution. This low-cost and plug-and-play tracking component showed to capture certain movements and exercises with an accuracy, that is sufficient for our rehabilitation context, while it was not so well or not at all suited for others. However, it can be expected that accuracy of markerless MoCap technologies will be improved in the future.

RGB-D sensors such as the Kinect require no markers or sensors on the human body and are easy to setup. Therefore, it can be hypothesized that they will play a vital role in the future of home-based games in rehabilitation.

Nevertheless, a single Kinect only covers a relatively small tracking space. Therefore, we have designed and developed a system consisting of several Kinects to assess the applicability in wide-area tracking scenarios. We distributed tracking data over network and merged poses in a flexible approach, that enabled us to almost arbitrarily scale the tracking volume. The evaluation showed, that combining multiple sensors worked for many of the assessed situations, although MoCap data still suffered from errors caused by the optical tracking principle (e.g. partial self-occlusion of the user).

Developing applications and serious games for rehabilitation, especially with a Virtual Reality (VR) setup, requires a lot of time and effort, because in addition to the implementation of game logic and content, often input/output devices, such as the above, have to be integrated and their data processed. Therefore, for serious games in rehabilitation and other VR applications in research and teaching we have developed a powerful framework - ARTiFICe -, that is lightweight and flexible. Furthermore, we integrated several new devices and technologies for tracking and biosignal acquisition. In addition, the framework incorporates modules for interaction, distribution and haptic feedback. These features together with the integration of an off-the-shelf game engine makes our framework very well suited for the development of serious games. The versatility of the ARTiFICe framework was demonstrated by the different VR environments, that were developed based on the framework using various hardware setups and interaction devices. These systems are using a wide variety of devices and offer different degrees of immersion.

Finally, the design and development of a serious game based on the ARTiFICe framework was described, that targets rehabilitation of patients with musculoskeletal chronic pain of the lower back and neck, a group that has previously been neglected by serious games. Medical requirements of the indication as well as key concepts of rehabilitation and design considerations and their integration in the game design were detailed.

A user study with a preliminary prototype of the game presented interesting insights and showed how important user feedback is in the game development process. Small details of the game, such as poor visibility of certain game elements or a poorly adjusted level of challenge in the game, have been identified to significantly reduce enjoyment and improved accordingly.

The final game prototype had been evaluated in a user study with a sample of ten adults with musculoskeletal pain in a training period of 4 weeks and the results presented in this thesis showed potential efficacy. Despite the short training period, overall the patients made a progression in terms of the requested motor skills in the mini-games during the 4 weeks of gaming, especially those patients with impaired motor skills. After 4 weeks of gaming, generally patients were capable of walking faster and reaching higher and experienced an increase in neck mobility. On a group level, the progression made on walking velocity is clinically relevant.

The game clearly motivated patients to perform their exercises, they enjoyed playing the game and liked the game environment and game play. Therefore, serious games in rehabilitation, such as ours, encourage patients to perform their exercises, and could have the potential to overcome the generally low compliance with home-based exercise programs.

Another positive aspect of our serious game is the availability of game data. These data

provide the therapist with detailed information on the progression of a patient in terms of the various trained motor skills. By using the available game data, the therapist can better align the game session to the needs of the individual patient. Furthermore, the transparency of the treatment for the patients can be increased, which matches the current trend in healthcare.

8.2 Open Issues

- **Improving the skeleton model:** The skeleton model introduced in our MoCap system is based on purely rotational joints with 3 degrees of freedom. During an extended calibration step the system could calculate constraints for the joints, which would reduce the amount of possible solutions during tracking. Furthermore, a translational component in the skeleton model's joints could be introduced to represent real human joints more accurately. Both measures could be used to gain either more robust or accurate tracking data in certain situations, or allow simplifications of the setup (e.g. reduce the number of markers required).

- **Extending evaluation of the MoCap system:** To acquire ground truth kinematic data of human MoCap data for evaluation is difficult and requires intrusive means and an enormous effort. However, a simulation with detailed kinematic models of the human joints, muscles and skin could provide datasets for an extended evaluation.

- **Multiuser low-cost wide area MoCap system:** Tracking multiple users can be interesting for a number of applications including serious games for rehabilitation. To extend our work to the tracking of multiple users, we would have to keep track of the global position of each user and identify each newly detected user in a cell. Furthermore, skeleton calibration data of the users has to be shared, which requires efficient transmission of that data over the network.

- **Elaborating pose merging strategies of the low-cost wide area MoCap system:** Combining data from multiple sensors could be improved through either merging data on a point cloud level or by application of more elaborate pose merging strategies. The earlier method should be more effective in certain situations (e.g. when merging depth data of two sensors each of which alone would be insufficient to produce a skeleton at all, such as combining two half images of a user's torso). This in turn could be used to increase the spacing between sensors or overall accuracy. However, calculating, transmitting and merging point clouds (and reprojection to a depth image required for the MoCap algorithm) would generate a substantial overhead. More elaborate pose merging strategies on the other hand could employ movement trends through, e.g. double exponential smoothing prediction, for combining poses. Furthermore, in absence of reliable confidence values of poses, they could be derived from the user's posture relative to the sensor (e.g. a sensor's view on the right shoulder will evidently produce low confidence of the left shoulder).

- **Extending haptic feedback:** Haptic indication for motion dynamics, such as velocity, showed to work well in our evaluation. In current work, building upon these findings,

we attempt to provide gesture training through haptic feedback. Using carefully designed vibrotactile stimuli and patterns we want to facilitate the learning phase required for users new to an interactive environment.

- **ARTiFICe as an open source framework:** We plan to release our framework as open source project to developers and the research community.
- **Extending the game for chronic pain rehabilitation:** The current version of the game only involves three mini-games, and an increase of that number would be beneficial. With more mini-games available, a game session could be better adjusted to the rehabilitation goals of the individual patient, and that patient's treatment protocol could be refined.
- **Adaptation of the game for remote physical rehabilitation:** In the current setting, patients were dependent on the availability of the therapist and the Roessingh Research and Development lab to play the game. Besides, an elaborate MoCap system was needed to generate input for the game. For a remote physical rehabilitation setting, there must be an easy-to-use and cheap alternative for this MoCap system, such as Microsoft's Kinect. Kinect was integrated into our game and tested as an alternative low-cost MoCap system. It allowed at least partial control of the mini-games by the requested motor skills in several cases. However, the mini-games and exercises would need to be adapted to the capabilities of the sensor and the sole control by the patient.
- **Extending the evaluation of the game:** A next step in the evaluation of our serious game is to compare it with conventional physiotherapy for patients suffering from chronic musculoskeletal pain and to assess them on clinical benefit, user experience, and costs.

  Concerning our game as a tool for remote physical rehabilitation, a next step is to evaluate the game in combination with low-cost maintenance-free motion capturing systems in a home setting and see if the outcome on accessibility and quality are comparable with the outcomes of this study. Our existing knowledge about telerehabilitation suggests that our game has the potential to increase the quality and accessibility of health-care and perhaps lower the costs.

Bibliography

[1] 3dconnexion. *SpaceNavigator*. 2015. URL: http://www.3dconnexion.de/products/spacemouse/spacenavigator.html (visited on 09/07/2015).

[2] Clark C Abt. *Serious games*. University Press of America, 1987.

[3] Gazihan Alankus, Amanda Lazar, Matt May, and Caitlin Kelleher. "Towards customizable games for stroke rehabilitation". In: *Proceedings of the 28th international conference on Human Factors in Computing Systems - CHI '10*. New York, New York, USA: ACM Press, 2010, pp. 2113–2122.

[4] Michael S Andersen. "Do kinematic models reduce the effects of soft tissue artefacts in skin marker-based motion analysis? An in vivo study of knee kinematics". In: *Journal of Biomechanics* 43.2 (2010), pp. 268–273.

[5] Mubashir Arain, Michael J Campbell, Cindy L Cooper, and Gillian A Lancaster. "What is a pilot or feasibility study? A review of current practice and editorial policy." In: *BMC medical research methodology* 10 (2010), p. 67.

[6] Andreas Aristidou and Joan Lasenby. "Real-time marker prediction and CoR estimation in optical motion capture". In: *The Visual Computer* 29.1 (2013), pp. 7–26.

[7] *Asus Xtion*. 2015. URL: http://www.asus.com/Multimedia/Xtion%5C_PRO/ (visited on 08/25/2015).

[8] ATS Committee on Proficiency Standards for Clinical Pulmonary Function Laboratories. "ATS statement: guidelines for the six-minute walk test". In: *American journal of respiratory and critical care medicine* 166.1 (2002), pp. 111–117.

[9] Seongmin Baek, Seungyong Lee, and Gerard Jounghyun Kim. "Motion retargeting and evaluation for VR-based training of free motions". In: *The Visual Computer* 19.4 (2003), pp. 222–242.

[10] Aaron Bangor, Philip Kortum, and James Miller. "Determining what individual SUS scores mean: Adding an adjective rating scale". In: *Journal of usability studies* 4 (2009), pp. 114–123.

[11] Kai Berger, Kai Ruhl, Yannic Schroeder, Christian Bruemmer, Alexander Scholz, and Marcus A Magnor. "Markerless Motion Capture using multiple Color-Depth Sensors." In: *Vision, Modeling, and Visualization*. 2011, pp. 317–324.

[12] Aimee L Betker, Ankur Desai, Cristabel Nett, Naaz Kapadia, and Tony Szturm. “Game-based exercises for dynamic short-sitting balance rehabilitation of people with chronic spinal cord and traumatic brain injuries.” In: *Physical Therapy* 87.10 (Oct. 2007), pp. 1389–98.

[13] Elaine Biddiss and Jennifer Irwin. “Active video games to promote physical activity in children and youth: a systematic review.” In: *Archives of pediatrics & adolescent medicine* 164 (2010), pp. 664–672.

[14] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. “The MagicBook: a Transitional AR Interface”. In: *Computers & Graphics* 25.5 (2001), pp. 745–753.

[15] Mark T Bolas. “Human factors in the design of an immersive display”. In: *Computer Graphics and Applications, IEEE* 14.1 (1994), pp. 55–59.

[16] *Boost*. 2015. URL: `www.boost.org` (visited on 08/25/2015).

[17] Doug A Bowman and Larry F Hodges. “An Evaluation of Techniques for Grabbing and Manipulating Objects in Immersive Virtual Environments Arm-Extension Ray-Casting”. In: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. 1997, pp. 35–38.

[18] Doug A Bowman, Ernst Kruijff, Joseph J LaViola Jr., and Ivan Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2004.

[19] J Brooke. “SUS – a quick and dirty usability scale”. In: *P.W. Jordan, B. Thomas, B.A. Weerdmeester et al. (Eds.), Usability Evaluation in Industry. London: Taylor & Francis.* (1995), pp. 189–194.

[20] R G A Bults, D F Knoppel, I A Widya, L Schaake, and H J Hermens. “The myofeedback-based teletreatment system and its evaluation”. In: *Journal of Telemedicine and Telecare* 16.6 (2010), pp. 308–315.

[21] Grigore Burdea. “Keynote address: Virtual rehabilitation-benefits and challenges”. In: *1st International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational) VRMHR*. sn. 2002, pp. 1–11.

[22] Grigore C Burdea and Philippe Coiffet. *Virtual Reality Technology*. Wiley-IEEE Press, 2003.

[23] J W Burke, M D J McNeill, D K Charles, P J Morrow, J H Crosbie, and S M McDonough. “Optimising engagement for stroke rehabilitation using serious games”. In: *The Visual Computer* 25.12 (2009), pp. 1085–1099.

[24] S R Buss and J S Kim. “Selectively damped least squares for inverse kinematics”. In: *Journal of Graphics, GPU, & Game Tools* 10.3 (2005), pp. 37–49.

[25] E H Calvillo-G´amez, P Cairns, and A L Cox. *Assessing the Core Elements of the Gaming Experience*. London, UK, 2010.

[26] Jonathan Cameron and Joan Lasenby. *A Real-Time Sequential Algorithm for Human Joint Localization*. 2005.

[27] Emiko Charbonneau, Andrew Miller, and Joseph J LaViola. “Teach me to dance: exploring player experience and performance in full body dance games”. In: *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE ’11*. Nov. 2011, 43:1–43:8.

[28] J. A. Cramer, Á Benedict, N. Muszbek, A. Keskinaslan, and Z. M. Khan. *The significance of compliance and persistence in the treatment of diabetes, hypertension and dyslipidaemia: A review*. 2008.

[29] Chris Crawford. *Chris Crawford on Game Design*. 2003.

[30] C Cruz-Neira, D J Sandin, and T A DeFanti. “Surround-Screen Projection-Based Virtual Reality: the Design and Implementation of the CAVE”. In: *20th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 1993, pp. 135–142.

[31] Carolina Cruz-Neira, Allen Bierbaum, Patrick Hartling, Christopher Just, and Kevin Meinert. “VR Juggler – An Open Source Platform for Virtual Reality Applications”. In: *Proceedings of IEEE Virtual Reality*. Reno, Nevada, USA: IEEE, 2001, pp. 89–96.

[32] Zeller. David. “Physics driven 3D Dynamic Geometry Software for Elementary Education”. Master’s thesis. Vienna University of Technology, 2012.

[33] H K DeChant, W G Tohme, S K Mun, W S Hayes, and K A Schulman. “Health systems evaluation of telemedicine: a staged approach.” In: *Telemedicine journal : the official journal of the American Telemedicine Association* 2 (1996), pp. 303–312.

[34] J Decker, H Li, D Losowyj, and V Prakash. “Wiihabilitation: rehabilitation of wrist flexion and extension using a wiimote-based game system”. In: *Governor’s School of Engineering and Technology Research Journal* (2009).

[35] Jonathan Deutscher and Ian Reid. “Articulated Body Motion Capture by Stochastic Search”. In: *International Journal of Computer Vision* 61.2 (2005), pp. 185–205.

[36] Marco Dozza, Fay B Horak, and Lorenzo Chiari. “Auditory biofeedback substitutes for loss of sensory information in maintaining stance.” In: *Experimental brain research* 178.1 (2007), pp. 37–48.

[37] David England. *Whole body interaction*. Springer, 2011.

[38] Jan B F Van Erp, Hendrik a. H C Van Veen, Chris Jansen, and Trevor Dobbins. “Waypoint navigation with a vibrotactile waist belt”. In: *ACM Transactions on Applied Perception*. Vol. 2. 2. 2005, pp. 106–117.

[39] Peter Fikar, Christian Schönauer, and Hannes Kaufmann. “The Sorcerer’s Apprentice: A serious game aiding rehabilitation in the context of Subacromial Impingement Syndrome”. In: *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*. IEEE, 2013, pp. 327–330.

[40] Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. “Aperture based Selection for Immersive Virtual Environments”. In: *Proceedings of the 9th ACM Symposium on User Interface Software & Technology*. 1996, pp. 95–96.

[41] Kenichiro Fukushi, Jan Zizka, and Ramesh Raskar. “Second Skin: Motion Capture with Actuated Feedback for Motor Learning”. In: *ACM SIGGRAPH 2011*. 2011, p. 51.

[42] Juergen Gall, Carsten Stoll, Edilson De Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans Peter Seidel. “Motion capture using joint skeleton tracking and surface estimation”. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*. 2009, pp. 1746–1753.

[43] TN Games. *3RD Space*. 2015. URL: `http://tngames.com/` (visited on 01/01/2015).

[44] M S Geroch. “Motion capture for the rest of us”. In: *Journal of Comp. Sciences in Colleges* 19.3 (2004), pp. 157–164.

[45] Georg Gerstweiler and Emanuel Vonach. “Development of an Active Motion Capture Suit for Teaching Motion Skills”. MA thesis. 188/2, 2011.

[46] A Gift. “Visual Analogue Scales: Measurement of subjective phenomena”. In: *Nursing Research* 38.5 (1989), pp. 286–287.

[47] Oonagh M Giggins, Ulrik McCarthy Persson, and Brian Caulfield. “Biofeedback in rehabilitation.” In: *Journal of Neuroengineering and Rehabilitation* 10.60 (2013).

[48] José-Antonio Gil-Gómez, Roberto Lloréns, Mariano Alcañiz, and Carolina Colomer. “Effectiveness of a Wii balance board-based system (eBaViR) for balance rehabilitation: a pilot randomized clinical trial in patients with acquired brain injury.” In: *Journal of neuroengineering and rehabilitation* 8.30 (2011).

[49] Lee Graves, Gareth Stratton, N D Ridgers, and N T Cable. “Comparison of energy expenditure in adolescents when playing new generation and sedentary computer games: cross sectional study.” In: *BMJ (Clinical research ed.)* 335 (2007), pp. 1282–1284.

[50] G.tec. *g.MOBIlab biosignal acquisition*. http://www.gtec.at. 2015. URL: `http://www.gtec.at` (visited on 08/25/2015).

[51] Gutemberg B Guerra-Filho1. “Optical Motion Capture: Theory and Implementation”. In: *Journal of Theoretical and Applied Informatics (RITA)* 12.2 (2005).

[52] M Hasenbring. “Attentional control of pain and the process of chronification”. In: *Progress in pain research* 129 (2000), pp. 525–534.

[53] M Hasenbring, G Marienfeld, D Kuhlendahl, and D Soyka. “Risk factors of chronicity in lumbar disc patients. A prospective investigation of biologic, psychologic, and social predictors of therapy outcome.” In: *Spine* 19 (1994), pp. 2759–2765.

[54] L Herda, P Fua, R Plänkers, R Boulic, and D Thalmann. “Skeleton-based motion capture for robust reconstruction of human motion”. In: *Computer Animation 2000 (CA’00)* (2000), p. 77.

[55] Gerd Hesina. “Distributed Collaborative Augmented Reality”. PhD thesis. Vienna University of Technology, 2001.

[56] H Hiroshi Kawasaki, R Furukawa, R Sagawa, and Yagi Yasushi. “Dynamic scene shape reconstruction using a single structured light pattern”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008, pp. 1–8.

[57] M K Holden. “Virtual environments for motor rehabilitation: review”. In: *CyberPsychology & Behavior* 8.3 (2005), pp. 187–211.

[58] Berthold K P Horn. “Closed-form solution of absolute orientation using unit quaternions”. In: *Journal of the Optical Society of America* 4 (1987).

[59] A Hornung, S Sar-Dessai, and L Kobbelt. “Self-calibrating optical motion tracking for articulated bodies”. In: *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*. 2005, pp. 75–82.

[60] Razer Inc. *Hydra*. 2015. URL: `http://www.razerzone.com/vrpromo/` (visited on 09/07/2015).

[61] Ali Israr and Ivan Poupyrev. “Tactile brush: Drawing on Skin with a Tactile Grid Display”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, May 2011, pp. 2019–2028.

[62] J. A. Jacko. *The Human Computer Interaction Handbook*. Third. CRC Press, 2012.

[63] Stephanie Jansen-Kosterink, Rianne M H A Huis in ’t Veld, Christian Schönauer, Hannes Kaufmann, Hermie J Hermens, and Miriam Vollenbroek-Hutten. “A Serious Exergame for Patients Suffering from Chronic Musculoskeletal Back and Neck Pain: A Pilot Study”. In: *Games for Health* 2.5 (2013), pp. 299–307.

[64] Bernhard Kainz, Stefan Hauswiesner, Gerhard Reitmayr, Markus Steinberger, Raphael Grasset, Lukas Gruber, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, and Dieter Schmalstieg. “OmniKinect: Real-Time Dense Volumetric Data Acquisition and Applications”. In: *Proceedings of the 18th ACM symposium on Virtual reality software and technology*. 2012, pp. 25–32.

[65] Hirokazu Kato and Mark Billinghurst. “Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System”. In: *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*. IEEE, 1999, pp. 85–94.

[66] Pamela M. Kato. “Video games in health care: Closing the gap.” In: *Review of General Psychology* 14 (2010), pp. 113–121.

[67] F J Keefe and R W Hill. “An objective approach to quantifying pain behavior and gait patterns in low back pain patients.” In: *Pain* 21 (1985), pp. 153–161.

[68] Anne Kennedy, Anne Rogers, and Peter Bower. “Support for self care for patients with chronic disease”. In: *BMJ : British Medical Journal* 335 (2007), pp. 968–970.

[69] Hadi Kharrazi, Amy Shirong Lu, Fardad Gharghabi, and Whitney Coleman. “A scoping review of health game research: Past, present, and future”. In: *Games for Health Journal* 1 (2012), pp. 153–164.

[70] Adam G Kirk, James F O\’Brien, and David A Forsyth. “Skeletal Parameter Estimation from Optical Motion Capture Data”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 Volume 2*. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2005.

[71] Monika Klapdohr, Björn Wöldecke, Dionysios Marinos, Jens Herder, Christian Geiger, and Wolfgang Vonolfen. “Vibrotactile Pitfalls: Arm Guidance for Moderators in Virtual TV Studios”. In: *Information Systems* (2010), pp. 72–80.

[72] Steffen Knoop, Stefan Vacek, Klaus Steinbach, and Rudiger Dillmann. “Sensor fusion for model based 3D tracking”. In: *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*. 2006, pp. 524–529.

[73] Peter Konrad. “The abc of emg”. In: *A practical introduction to kinesiological electromyography* 1 (2005).

[74] Raph Koster. *Theory of Fun for Game Design*. 2005, p. 256.

[75] Jan Kowalczewski, Su Ling Chong, Mary Galea, and Arthur Prochazka. “In-home telerehabilitation improves tetraplegic hand function.” In: *Neurorehabilitation and neural repair* 25 (2011), pp. 412–422.

[76] Roland Kuck, Jürgen Wind, Kai Riege, and Manfred Bogen. “Improving the AVANGO VR/AR Framework - Lessons Learned”. In: *5th Workshop of the GI-VR/AR Group*. Magdeburg, Germany: VDTC, 2008.

[77] Gert Kwakkel. “Intensity of practice after stroke: More is better”. In: *Schweizer Archiv fur Neurologie und Psychiatrie* 160 (2009), pp. 295–298.

[78] B. Lange, Sheryl M. Flynn, and A. A. Rizzo. “Game-based telerehabilitation”. In: *European Journal of Physical and Rehabilitation Medicine* 45 (2009), pp. 143–151.

[79] Belinda Lange, Chien-Yen Chang, Evan Suma, Bradley Newman, Albert Skip Rizzo, and Mark Bolas. “Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor”. In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE. 2011, pp. 1831–1834.

[80] Belinda Lange, Sebastian Koenig, Chien-Yen Chang, Eric McConnell, Evan Suma, Mark Bolas, and Albert Rizzo. “Designing informed game-based rehabilitation tasks leveraging advances in virtual reality”. In: *Disability and rehabilitation* 34.22 (2012), pp. 1863–1870.

[81] Joseph J LaViola. “Double exponential smoothing: an alternative to Kalman filter-based predictive tracking”. In: *EGVE ’03: Proceedings of the workshop on Virtual environments 2003*. New York, NY, USA: ACM, 2003, pp. 199–206.

[82] Jeff Lieberman and Cynthia Breazeal. “TIKL: Development of a Wearable Vibrotactile Feedback Suit for Improved Human Motor Learning”. In: *IEEE Transactions on Robotics* 23.5 (Oct. 2007), pp. 919–926.

[83] Robert W. Lindeman, Yasuyuki Yanagida, Haruo Noma, and Kenichi Hosaka. “Wearable vibrotactile systems for virtual contact and information display”. In: *Virtual Reality* 9.2-3 (2006), pp. 203–213.

[84] W T Liu, C H Wang, H C Lin, S M Lin, K Y Lee, Y L Lo, S H Hung, Y M Chang, K F Chung, and H P Kuo. “Efficacy of a cell phone-based exercise programme for COPD”. In: *Eur Respir J.* 32.3 (2008), pp. 651–659.

[85] Minhua Ma and Kamal Bechkoum. “Serious games for movement therapy after stroke”. In: *2008 IEEE International Conference on Systems, Man and Cybernetics*. Oct. 2008, pp. 1872–1877.

[86] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. “DART: a Toolkit for Rapid Design Exploration of Augmented Reality Experiences”. In: *Proceedings of the 17th ACM Symposium on User Interface Software and Technology*. ACM Publications, 2004, pp. 197–206.

[87] Niall Maclean, Pandora Pound, Charles Wolfe, and Anthony Rudd. “The concept of patient motivation a qualitative analysis of stroke professionals attitudes”. In: *Stroke* 33.2 (2002), pp. 444–448.

[88] Simon McCallum. “Gamification and serious games for personalized health”. In: *Studies in Health Technology and Informatics*. Vol. 177. 2012, pp. 85–96.

[89] Troy McDaniel, Daniel Villanueva, Sreekar Krishna, and Sethuraman Panchanathan. “MOVeMENT : A Framework for Systematically Mapping Vibrotactile Stimulations to Fundamental Body Movements”. In: *Computing* (2010), pp. 1–6.

[90] David R Michael and Sandra L Chen. *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade, 2005.

[91] Microsoft. *Kinect full body interaction*. http://www.xbox.com/kinect. 2015. URL: `http://www.xbox.com/en-US/kinect` (visited on 08/25/2015).

[92] Annette Mossel. “Robust Wide-Area Tracking and Intuitive 3D Interaction for Mixed Reality Environments”. PhD thesis. 2014.

[93] Annette Mossel, Christian Schönauer, Georg Gerstweiler, and Hannes Kaufmann. “ARTiFICe - Augmented Reality Framework for Distributed Collaboration”. In: *The International Journal of Virtual Reality* 11.3 (2012), pp. 1–7.

[94] *Motion Analysis*. 2015. URL: `http://www.motionanalysis.com/` (visited on 08/25/2015).

[95] Akio Nakamura, Sou Tabata, Tomoya Ueda, Shinichiro Kiyofuji, and Yoshinori Kuno. “Dance training system with active vibro-devices and a mobile image display”. In: *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3075–3080.

[96] Andrei Ninu. “Prosthesis Embodiment : Sensory-Motor Integration of Prosthetic Devices into the Amputee’s Body Image”. PhD Thesis. Vienna University of Technology, 2013.

[97] Stepan Obdrzalek, Gregorij Kurillo, Ferda Ofli, Ruzena Bajcsy, Edmund Seto, Holly Jimison, and Misha Pavel. “Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population”. In: *Engineering in medicine and biology society (EMBC), 2012 annual international conference of the IEEE*. IEEE. 2012, pp. 1188–1193.

[98] J F O'Brien, R E Bodenheimer Jr, G J Brostow, and J K Hodgins. "Automatic joint parameter estimation from magnetic motion capture data". In: *In Proceedings of Graphics Interface* (2000), pp. 53–60.

[99] *Oculus VR*. 2015. URL: www.oculus.com (visited on 08/25/2015).

[100] *Opentracker 2.0*. URL: https://www.ims.tuwien.ac.at/projects/opentracker2 (visited on 08/25/2015).

[101] *OpenVideo*. URL: http://rpm.icg.tugraz.at/.

[102] P R J Östergard. "A fast algorithm for the maximum clique problem". In: *Journal of Discrete Applied Mathematics* 120.1-3 (2002), pp. 197–207.

[103] *Perception Neuron MOCAP*. 2015. URL: https://neuronmocap.com/ (visited on 08/25/2015).

[104] PhaseSpace. *PhaseSpace Motion Capture*. 2015. URL: http://www.phasespace.com (visited on 08/25/2015).

[105] Thomas Pintaric and Hannes Kaufmann. "Affordable Infrared-Optical Pose-Tracking for Virtual and Augmented Reality". In: *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments Workshop*. IEEE VR 2007. 2007, pp. 44–51.

[106] C Plagemann, V Ganapathi, D Koller, and S Thrun. "Real-time identification and localization of body parts from depth images". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010, pp. 3108–3113.

[107] Stefan Posland. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. John Wiley & Sons, 2009.

[108] Ivan Poupyrev and Mark Billinghurst. "The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR". In: *Proceedings of the 9th annual ACM symposium on User Interface Software and Technology*. 1996, pp. 79–80.

[109] PrimeSense. *NITE*. (Due to acquisition of Primesense through Apple the web page is offline). URL: http://www.primesense.com/ (visited on 2011).

[110] Primesense. *OpenNI*. 2014. URL: http://structure.io/openni (visited on 08/25/2015).

[111] Qualcomm Inc. *Vuforia SDK*. 2015. URL: www.qualcomm.com/products/vuforia (visited on 08/25/2015).

[112] P Rego, P M Moreira, and L P Reis. "Serious games for rehabilitation: A survey and a classification towards a taxonomy". In: *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*. 2010, pp. 1–6.

[113] RehaCom. *RehaCom cognitive rehabilitation*. 2015. URL: http://www.rehacom.com (visited on 08/25/2015).

[114] G Reitmayr and D Schmalstieg. "An open software architecture for virtual reality interaction". In: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM. 2001, pp. 47–54.

[115] Maurice Ringer and Joan Lasenby. “A Procedure for Automatically Estimating Model Parameters in Optical Motion Capture”. In: *Image and Vision Computing* 22.10 (2002).

[116] Albert Rizzo, Arno Hartholt, Mario Grimani, Andrew Leeds, and Matt Liewer. “Virtual Reality Exposure Therapy for Combat-Related Posttraumatic Stress Disorder”. In: *Computer* 47.7 (2014), pp. 31–37.

[117] Jacob Rosenthal, Nathan Edwards, Daniel Villanueva, Sreekar Krishna, Troy McDaniel, and Sethuraman Panchanathan. “Design, Implementation, and Case Study of a Pragmatic Vibrotactile Belt”. In: *IEEE Trans. on Instrumentation and Measurement* 60.1 (2011), pp. 114–125.

[118] Jean-Sébastien Roy, Hélène Moffet, and Bradford J McFadyen. “The effects of unsupervised movement training with visual feedback on upper limb kinematic in persons with shoulder impingement syndrome.” In: *Journal of electromyography and kinesiology : Official journal of the International Society of Electrophysiological Kinesiology* 20.5 (Oct. 2010), pp. 939–46.

[119] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT press, 2004.

[120] Jonas Sandlund, Ulrik Röijezon, Martin Björklund, and Mats Djupsjöbacka. “Acuity of goal-directed arm movements to visible targets in chronic neck pain”. In: *Journal of Rehabilitation Medicine* 40 (2008), pp. 366–374.

[121] Gustavo Saposnik, Robert Teasell, Muhammad Mamdani, Judith Hall, William McIlroy, Donna Cheung, Kevin E. Thorpe, Leonardo G. Cohen, and Mark Bayley. “Effectiveness of virtual reality using wii gaming technology in stroke rehabilitation: A pilot randomized clinical trial and proof of principle”. In: *Stroke* 41 (2010), pp. 1477–1484.

[122] I Schiller. *MIP - MultiCameraCalibration*. 2015. URL: `http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Calibration` (visited on 08/25/2015).

[123] Dieter Schmalstieg, Anton Fuhrmann, Gerd Hesina, Zsolt Szalavári, Miguel Encarnacao, Michael Gervautz, and Werner Purgathofer. “The Studierstube augmented reality project”. In: *Presence - Teleoperators and Virtual Environments* 11.1 (2002), pp. 33–54.

[124] R A Schmidt and T D Lee. *Motor control and learning: A behavioral emphasis*. 4th ed. Human Kinetics Publishers, 2005.

[125] Christian Schönauer. “Skeletal Structure Generation for Optical Motion Capture.” Diploma thesis. Vienna University of Technology, 2007.

[126] Christian Schönauer, Kenichiro Fukushi, Alex Olwal, Hannes Kaufmann, and Ramesh Raskar. “Multimodal Motion Guidance: Techniques for Adaptive and Dynamic Feedback”. In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. New York, NY, USA: ACM, 2012, pp. 133–140.

[127] Christian Schönauer and Hannes Kaufmann. “Wide Area Motion Tracking Using Consumer Hardware”. In: *The International Journal of Virtual Reality* 12.1 (2013), pp. 1–9.

[128] Christian Schönauer, Hannes Kaufmann, Stephanie Jansen-Kosterink, and Miriam Vollenbroek-Hutten. “Design eines Serious Games für die Rehabilitation von chronischen Rückenschmerzen”. In: *Future and Reality of Gaming Vienna Games Conference, FROG 2012, Game Over. Was nun? Vom Nutzen und Nachteil des digitalen Spiels für das Leben*. Bundesministerium für Wirtschaft, Familie und Jugend, Abt. II/5, 2012, pp. 31–46.

[129] Christian Schönauer, Thomas Pintaric, and Hannes Kaufmann. “Full body interaction for serious games in motor rehabilitation”. In: *Proceedings of the 2nd Augmented Human International Conference*. AH ’11. New York, NY, USA: ACM, 2011, 4:1–4:8.

[130] Christian Schönauer, Thomas Pintaric, and Hannes Kaufmann. “Full Body Motion Capture - A Flexible Marker Based Solution”. In: *Proceedings of Workshop on Accessibility Engineering with user models, simulation and VR*. 2012.

[131] Christian Schönauer, Thomas Pintaric, Hannes Kaufmann, Stephanie Jansen-Kosterink, and Miriam Vollenbroek-Hutten. “Chronic Pain Rehabilitation with a Serious Game using Multimodal Input”. In: *Proceedings of Virtual Rehabilitation 2011*. 2011, pp. 1–8.

[132] E Schuurink, J Houtkamp, and A Toet. “Engagement and EMG in Serious Gaming: Experimenting with Sound and Dynamics in the Levee Patroller Training Game”. In: *Fun and Games* (2008), pp. 139–149.

[133] Robert Sedgewick. *Algorithms*. Ed. by Addison-Wesley. Addison-Wesley Publishing Co, 1984.

[134] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Aug. 1994. URL: `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`.

[135] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. “Real-time Human Pose Recognition in Parts from Single Depth Images”. In: *Communications of the ACM* 56.1 (Jan. 2013), pp. 116–124.

[136] M.-C. Silaghi, R Plankers, R Boulic, P Fua, and D Thalmann. “Local and Global Skeleton Fitting Techniques for Optical Motion Capture”. In: *IFIP CapTech 98, Geneva*. Lecture Notes in Artificial Intelligence (1998), pp. 26–40.

[137] Per Sjölander, Peter Michaelson, Slobodan Jaric, and Mats Djupsjöbacka. “Sensorimotor disturbances in chronic neck pain - range of motion, peak velocity, smoothness of movement, and repositioning acuity”. In: *Manual Therapy* 13 (2008), pp. 122–131.

[138] *SoftKinetic*. 2015. URL: `http://www.softkinetic.com/` (visited on 08/25/2015).

[139] Daniel Spelmezan, Mareike Jacobs, Anke Hilgers, and Jan Borchers. “Tactile motion instructions for physical activities”. In: *Proceedings of the 27th international conference on Human factors in computing systems - CHI ’09* (2009), p. 2243.

[140] C D Spenkelink, M M R Hutten, H J Hermens, and B O L Greitemann. “Assessment of activities of daily living with an ambulatory monitoring system: a comparative study in patients with chronic low back pain and nonsymptomatic controls.” In: *Clinical rehabilitation* 16 (2002), pp. 16–26.

[141] Kay M. Stanney, Ronald R. Mourant, and Robert S. Kennedy. “Human Factors Issues in Virtual Environments: A Review of the Literature”. In: *Presence: Teleoperators and Virtual Environments* 7.4 (1998), pp. 327–351.

[142] B G Steele, B Belza, K C Cain, J Coppersmith, S Lakshminarayan, J Howard, and J KHaselkorn. “A randomized clinical trial of an activity and exercise adherence intervention in chronic pulmonary disease”. In: *Arch Phys Med Rehabil.* 89 (2008), pp. 404–412.

[143] Evan Suma, David Krum, Belinda Lange, Skip Rizzo, and Marc Bolas. “FAAST: The Flexible Action and Articulated Skeleton Toolkit.” In: *Proceeding of IEEE Virtual Reality*. Costa Mesa, USA: IEEE, 2012, pp. 247–248.

[144] D Tabakin and C L Vaughan. “A comparison of 3D gait models based on the Helen Hayes marker set”. In: *Proceedings of the sixth international symposium on the 3D analysis of human movement*. 2000, pp. 98–101.

[145] R C Tait, J T Chibnall, and S Krause. “The Pain Disability Index: psychometric properties”. In: *Pain* 40.2 (1990), pp. 171–182.

[146] Matthew J. D. Taylor, Darren McCormick, Teshk Shawis, Rebecca Impson, and Murray Griffin. “Activity-promoting gaming systems in exercise and rehabilitation”. In: *The Journal of Rehabilitation Research and Development* 48.10 (2011), pp. 1171–1186.

[147] Russel Taylor, Thomas C Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T Helser. *VRPN: a device-independent, network-transparent VR peripheral system*. Baniff, Alberta, Canada, 2001.

[148] The Captury GmBH. *The Captury Motion Capture*. 2015. URL: `http://www.thecaptury.com` (visited on 08/25/2015).

[149] TMSI. *Mobi biosignal acquisition*. 2015. URL: `http://www.tmsi.com/` (visited on 08/25/2015).

[150] Roy Tranberg, Tuuli Saari, Roland Zügner, and Johan Kärrholm. “Simultaneous measurements of knee motion using an optical tracking system and radiostereometric analysis (RSA)”. In: *Acta Orthopaedica* 82.2 (2011), pp. 171–176.

[151] Unity-Technologies. *Unity3D game engine*. http://unity3d.com/. 2015. URL: `http://unity3d.com/` (visited on 08/25/2015).

[152] M G Van Weering, M M Vollenbroek-Hutten, T M Tönis, and H J Hermens. “Daily physical activities in chronic lower back pain patients assessed with accelerometry”. In: *Eur J Pain* 13.6 (2009), pp. 649–654.

[153] Vicon. *Vicon motion capture system*. 2015. URL: `http://www.vicon.com` (visited on 08/25/2015).

[154] Virtools. *Virtools Dev User Guide*. 2015. URL: `http://www.virtools.com` (visited on 08/25/2015).

[155] J W Vlaeyen, A M Kole-Snijders, and R G Boeren. “Fear of movement/(re)injury in chronic low back pain and its relation to behavioral performance”. In: *Pain* 62.3 (1995), pp. 362–363.

[156] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. *Practical motion capture in everyday surroundings*. 2007.

[157] G E Voerman, M M R Vollenbroek-Hutten, L Sandsjö, R Kadefors, and H J Hermens. "Prognostic factors for the effects of two interventions for work-related neck-shoulder complaints: Myofeedback training and ergonomic counselling". In: *Applied Ergonomics* 39.6 (2008), pp. 743–753.

[158] Daniel Wagner and Dieter Schmalstieg. "ARToolKitPlus for Pose Tracking on Mobile Devices". In: *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*. Ed. by Michael Grabner and Helmut Grabner. 2007, pp. 139–146.

[159] Jia Wang and Robert W. Lindeman. *Unity Indie VRPN Adapter (UIVA)*. URL: `http://web.cs.wpi.edu/~gogo/hive/UIVA/` (visited on 08/25/2015).

[160] Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. "Realtime Performance-based Facial Animation". In: *ACM Transactions on Graphics (Proceedings SIGGRAPH 2011)*. 2011.

[161] Greg Welch and Eric Foxlin. "Motion Tracking: No Silver Bullet, but a Respectible Arsenal". In: *IEEE Computer Graphics and Applications* 22 (2002), pp. 24–38.

[162] Andrew D. Wilson and Hrvoje Benko. "Combining multiple depth cameras and projectors for interactions on, above and between surfaces". In: *Proceedings of the 23nd annual ACM symposium on User Interface Software and Technology - UIST '10* (2010), pp. 273–282.

[163] Astrid Woodhouse and Ottar Vasseljen. "Altered motor control patterns in whiplash and chronic neck pain." In: *BMC musculoskeletal disorders* 9.90 (2008).

[164] Key Words. "Philadelphia Panel Evidence-Based Clinical Practice Guidelines on for Neck Pain". In: *Physical Therapy* 81 (2001), pp. 1629–1640.

[165] *Xsens*. 2015. URL: `http://www.xsens.com/` (visited on 08/25/2015).

List of Figures

List of Tables