Lukas Rogl, BSc

The SquatController: A rehab training tool for tracking knee bend exercises with the Kinect

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. , Reinhold Scherer

Institute of Neural Engineering
Laboratory for Brain-Computer Interfaces

Head: Assoc. Prof. Dipl.-Ing. Dr.techn. Gernot Müller-Putz

Graz, August 2016

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

9.08.2016 .... [illegible]

date (signature)

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am 9.08.2016 .... [illegible]

(Unterschrift)

ACKNOWLEDGMENTS

I would like to use this opportunity to express my sincere gratitude to all people who supported and/or assisted me.

First of all, I want to thank my supervisor Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Reinhold Scherer, who suggested this interesting topic and always took time to answer my questions and helped me whenever some challenges showed up.

Furthermore I want to thank the people at the Institute of Neural Engineering, who always offered their assistance and provided valuable suggestions.

I want to express a special gratitude to Prim. Prof. Dr. Peter Grieshofer from the rehabilitation clinic in Judendorf Strassengel. The elaboration of the underlying conditions of a correct performed knee bend, which was essential in this thesis, was only possible with his assistance and the help of two of his associates, Birgit Mittermayr and Barbara Straßnig.

Finally I want to thank my whole family, my mother, father and my siblings and also my dear friends an colleagues for their support and encouragement during my studies.

Abstract

About every 21 minutes a stroke incident occurs in Austria. While lethal stroke incidents are continually decreasing, the negative side effects of strokes like postural asymmetry still remain. As a result of postural asymmetries, victims of a hemiplegic stroke are especially at risk of falling. In order to improve the symmetrical weight distribution and thus decreasing the risk of falling, particularly the training with high-intensity and repetitive task-specific practices yielded favorable results. Such motor rehabilitation training requires professional instructions and supervision by a specialist. However, if someone is not able to come to the clinic, exercises can also be trained by means of tele-rehabilitation. Such a tele-rehabilitation system has to be able to track the motions of the participants. The fundamental idea of improving the symmetrical weight distribution with the help of repetitive rehab exercises and the advantages of a tele-rehabilitation system, led to the consideration of creating a system with the ability to track a person and provide reliable feedback, while doing specific rehab exercises. Aim of this thesis is the creation of a software solution, called SquatController, for a rehab based training tool in order to track a knee bend exercise with the Kinect sensor (v2). The question of research is whether the new developed system is able to reliably track the knees during exercise performance and provide corresponding feedback based on the conditions of a physiotherapeutic correct knee bend. Furthermore, the usability of the specific software was questioned. A total number of 6 healthy subjects were measured. Each subject had to undergo 3 runs, consisting of 20 knee bends. With an averaged total score of 88,33 out of 100 points, regarding a performed usability evaluation, the developed SquatController software proved to be easy to handle, if initial instructions were provided. Additional examinations of the 4 used feedback parameters of the SquatController (Area0, Area2, Area4, Depth) confirmed the ability of the system to track the knees during exercise performance and provide adequate visual feedback. However, in order to bring the SquatController software to the next level, minor source modifications and further tests with more participants are advised.

Keywords: Kinect4Rehab, Kinect v2, tele rehabilitation, BCI Graz, knee bend, squat

Kurzfassung

Rund alle 21 Minuten erleidet eine Person in Österreich einen Schlaganfall - mit steigender Tendenz. Durch diesen Anstieg gibt es immer mehr Patienten mit Folgeschäden, wie zum Beispiel Asymmetrien in der Körperhaltung. Aufgrund solcher Asymmetrien sind speziell Patienten mit Hemiparese oder halbseitiger Lähmung besonders gefährdet zu fallen. Um eine Verbesserung der Gewichtsverteilung und somit eine Verringerung des Fallrisikos zu erzielen, stellte sich vor allem aufgabenspezifisch-repetitives und intensives Training als wirksam heraus. Solch ein Rehabilitationstraining benötigt eine professionelle Einführung und Überwachung. Ist es für eine Person jedoch zu anstrengend, eine Reha-Klinik aufzusuchen, gibt es auch die Möglichkeit, Trainingseinheiten mit Hilfe von Telerehabilitation durchzuführen. Diese Telerehabilitationssysteme müssen die Fähigkeit besitzen, die Bewegungen des Patienten im dreidimensionalen Raum zu überwachen (Motion Tracking). Die grundsätzliche Idee, anhand von repetitiven Reha-Übungen die symmetrische Verteilung des Körpergewichtes zu verbessern und zusätzlich dies mit einem Telerehabilitationssystem zu kombinieren, führte zu der Überlegung, ein System zu erzeugen, welches im Stande ist, eine physiotherapeutische Übung zu erkennen und zeitgleich Angaben bezüglich der Korrektheit anhand eines visuelles Feedbacks zu liefern. Ziel dieser Arbeit ist es, eine Softwarelösung zum Training in der Rehabilitation zu kreieren, welche mit Hilfe des Kinect Sensors eine Kniebeuge überwacht. Anhand dieser Arbeit sollen folgende wissenschaftliche Fragen beantwortet werden: 1) Ist es dem neu entwickelten System möglich, die Knie während einer Kniebeuge zuverlässig zu erkennen und simultan ein darauf basierendes Feedback zur Verfügung zu stellen? 2) Ist die neu entwickelte Software intuitiv und anwenderfreundlich? Insgesamt wurden 6 gesunde Personen vermessen. Pro Proband wurden 3 Messungen, bestehend aus je 20 Kniebeugen, durchgeführt. Die durchschnittliche Punktezahl von 88,33 von insgesamt 100 Punkten des durchgeführten Usability-Tests der SquatController-Software überschritt anfängliche Erwartungen. Zusätzliche Untersuchungen der 4 gewählten Feedbackparametern des SquatControllers (Area0, Area2, Area4, Depth) bestätigten dessen Fähigkeit, die Knie während der Durchführung von Kniebeugen zu überwachen und ein adäquates Feedback zu liefern. Um die Software in die nächste Phase zu bringen, ist es jedoch empfohlen, vereinzelte Modifizierungen des Quellcodes vorzunehmen und weitere Studien mit einer größeren Probandenzahl durchzuführen.

Schlüßelwörter: Kinect4Rehab, Kinect v2, Telerehabilitation, BCI Graz, Kniebeuge

Contents

List of Figures

List of Tables

Symbols, Abbreviations

Abbreviation	**Meaning**
SDK	Software Development Kit
ROI	Region of Interest
MVVM	Model - View - ViewModel
CPU	Central Processing Unit
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
USB	Universal Serial Bus
PC	Personal Computer
IR	Infrared
GUI	Graphic User Interface
FOV	Field of View

1. Introduction

According to the latest report of the Austrian Federal Ministry of Health [11] in 2015, about 25000 strokes (approx. 20 000 ischemic & 5 000 hemorrhagic) were registered within a year in Austria (status 2011, with rising trend). In other words, about every 21 minutes a stroke incident occurs in Austria. About 1200 of the ascertained strokes were lethal, however recordings over the past 10 years (2001 -2011) showed an annual decrease of 3.3% regarding the strokes with lethal ending.

The decrease of mortality can be ascribed to an earlier detection, a quicker intervention and a countrywide coverage of specialized stroke treatment facilities also known as stroke units. While the lethal stroke incidents are decreasing, the negative side effects of strokes still remain. Due to the fact that the brain literally controls the whole human body, these handicaps can occur in a variety of ways. Among many others numbness, speech disorder, limitations of the field of vision or postural asymmetry are possible candidates.

As a result of postural imbalances, victims of a hemiplegic stroke are especially at risk of falling. Nyberg and Gustafson [25], who investigated the incidence of falls among stroke patients, found that 37.2 % of 153 registered falls occurred during transfers, or while they were changing position from standing to sitting, or vice versa.

A systematic review done by Langhorne et al. [16] aimed on providing an overview of the available evidence on interventions for motor recovery after stroke. They concluded that, although the existing evidence is limited by poor trial designs, some treatments do show promise for improving motor recovery, particularly those that have focused on high-intensity and repetitive task-specific practice.

An investigation done by Cheng P. T. [8] examined the effect of symmetrical bodyweight distribution training in preventing falls among patients with hemiplegic stroke. The training was performed with the help of a special biofeedback trainer where the patient had to rise up and sit down, on an adjustable chair, as symmetrically as possible, while standing on a dual force platform. The evaluations yielded a significant improvement in sit-to-stand performance regarding the patients of the training group. Body weight was distributed more symmetrically in both legs, with less mediolateral sway in the center of pressure (COP) when rising and sitting down.

The execution of such motor rehabilitation training involves comprehensive, repetitive range of motion and coordination exercises [17]. These exercises require professional instructions, supervision and a critical evaluation of the patient's progress by a specialist. If someone wants to train at home on its own, in order to improve the functional motor outcome due to additional repetitive practice, the incorrect execution of a therapeutic exercise may not lead to the desired outcome or even worse, lead to some negative side effects. Therefore a guarantee of correctness is advised if someone wants to practice at home.

This is one of the reasons why the usage of telemedical based applications is continually growing since the 20^{th} century [3]. The so called tele-rehabilitation, one of the various subcategories, utilizes telecommunication networks and the Internet to provide rehabilitation services. The usage of such systems yields several advantages. One of these is the possibility to treat people who are not able to come to the clinic on their own.

In order to provide such a tele-rehabilitation system for training motor tasks, capturing three dimensional environments and objects (often referred to as motion tracking) is a necessary task. Human motion recognition technologies have been used to monitor physical rehabilitation exercises long before the release of Microsoft Kinect. However, most of them rely on motion tracking tools that are intrusive because patients either have to wear markers or attach inertial sensors [17]. Due to the big advantage of a marker less sensing technology and the additional low costs of about 200 $, the Kinect sensor was used as a tracking device. The sensor will be further introduced in chapter 1.1.

The fundamental idea of improving the sit-to-stand performance and enhance a more symmetrically weight distribution in order to decrease the amount of falls of stroke patients, as well as the advantages provided by tele-rehabilitation applications, led to the consideration of creating a system with the ability to track a person and provide reliable feedback, while doing some rehab exercises. Such a system would be a perfect training tool for therapists to coordinate home-based training sessions.

For this reason the thesis aims at creating a software solution for a rehab based training tool in order to track a knee bend or squat exercise with the Kinect sensor. Such a knee bend resembles the sit-to-stand performance of the mentioned investigation in the best way.

The question of research is whether the system is able to reliably track the knees during exercise performance and provide corresponding feedback based on a correct physiotherapeutic knee bend which is described in chapter 1.3. Furthermore, it was questioned whether the software is intuitive and easy to use.

1.1. The Microsoft Kinect sensor

Since the launch of Microsoft Kinect for Windows sensor v1 (Nov. 2010), the capabilities of low cost depth camera technologies have increased. With this particular sensor the door was opened for a low-cost alternative in the field of motion capturing. In addition, there is no need for special markers or a complex sensor setup due to the depth-sensing technique based on triangulation with structured light. With those advantages comes a certain lag of accuracy and precision, which have caused difficulties in the past few years regarding approaches of Kinect based rehab applications [13]. Recently, Microsoft released the second version of the Kinect sensor. Based on the revised sense technology (time of flight) it provides an improved depth measurement accuracy.

1.1.1. An overview of the history of the Kinect

The Kinect sensor like it is known today was firstly announced on June 1, 2009, under the project name ”Natal”. Later on it was renamed into Kinect, a portmanteau of the words ”kinetic” and ”connect”, which describe key aspects of the initiative.

It was launched in 2010 and became one of the most popular game controllers. More than 24 million units were sold as of February 2013 [1].

On February 5, 2012, Microsoft released the commercial version of Kinect for Windows SDK v1.0. With this software development kit a path was granted for the development of sophisticated computer-based human motion tracking applications in various programming languages like C# and C++.

A first standalone version of the second generation of the sensor named Kinect for Xbox One was released in October 2014. Prior to that, the Kinect was just included in the Xbox One bundles beginning with November 22, 2013. [2]

On July 15, 2014, a windows-compatible version of the new Kinect sensor with the proper SDK (2.0) was released. Microsoft additionally released an adapter, that allows to connect the originally Kinect for Xbox One sensor with the PC via USB 3.0 in October 2014, which further led to a discontinue of the actual Windows v2 product.

[1] obtained on May, 2016, from `http://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/`

[2] obtained on May, 2016, from `http://web.archive.org/web/20140702095342/http://www.computerandvideogames.com/463449/microsoft-to-release-xbox-one-without-kinect/`

1.1.2. A comparison between Kinect sensor version 1 and version 2

As mentioned before the SquatController software was initially intended to be realized with the first version of the Kinect sensors. Due to the increased depth accuracy, the greater field of view and a decreased influenceability by ambient light, the software was adapted for the second version of the Kinect. Table 1.1 summarizes the main differences between both versions of the Kinect sensor.

Table 1.1: Comparison of main features of the two versions of the Kinect sensor (aquired from [17]).

Feature	Kinect v1	Kinect v2
Depth Sensing Technology	Triangulation with structured light	Time of flight
Color Image Resolution	640x480 30fps 1280x960 12fps	1920x1080 30fps (12fps low light)
IR Image Resolution	640x480 30fps	512x424 30fps
Depth Sensing Resolution	640x480 30fps 320x240 30fps 80x60 30fps	512x424 30fps
Field of View	43° vertical 57° horizontal	> 43° vertical 70° horizontal
Depth Sensing Range	0.4m - 3m (near mode) 0.8m - 4m (default mode)	0.5m - 4.5m Up to 8m without skeletonization
Skeleton Tracking (with full skeleton)	Up to 2 subjects 20 joints per skeleton	Up to 6 subjects 25 joints per skeleton
Built-in Gestures	None	Hand state (open, close, lasso) Hand pointer controls; lean
Unity Support	Third party	Yes
Face APIs	Basic	Extended massively
Runtime Design	Can run multiple Kinect sensors per computer; One app per Kinect	At most one Kinect per computer; Multiple apps share same Kinect
Windows Store	Cannot publish to	Yes

1.1.3. A quick look inside the Kinect v2

A closer look at the inside of the sensor (see figure 1.1) reveals the locations of the RGB camera, the three IR-emitters with the corresponding depth sensor and the four microphones. On the back of the sensor a fan is used for cooling the electronic parts.

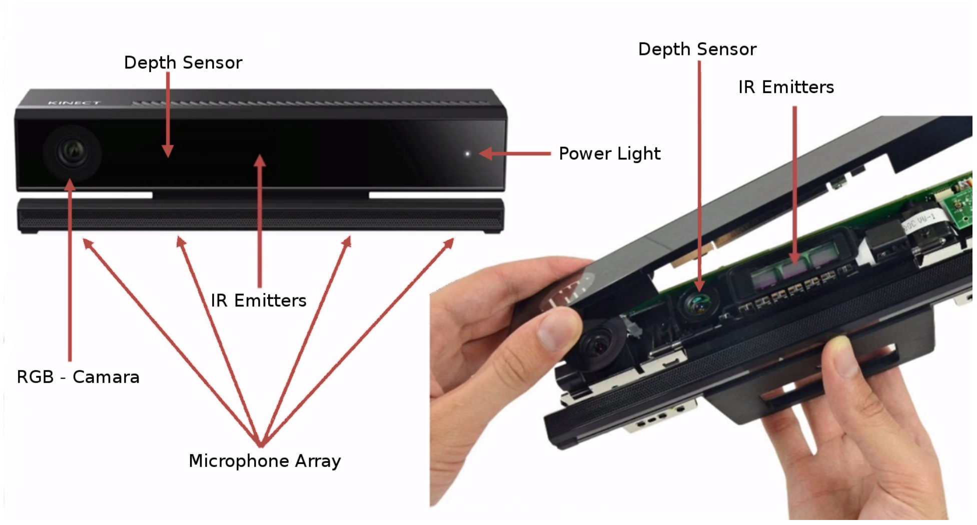


Figure 1.1: Inner parts of Kinect v2 (acquired from [12])

1.1.4. The depth sensing technology of the Kinect v2

The used depth sensing technology of the Kinect v2 is based on the time of flight method. The time of flight method usually measures the duration that it takes for an object, particle, electromagnetic or other wave to travel a certain distance through a medium. Thus for instance it can be used for measuring path lengths or velocities through a given medium. It can also be used for investigating the composition or flow rate of the particle or medium. Known applications which use the time of flight approach are for example near infrared spectroscopy and planar doppler velocimetry.

A time of flight camera is a range imaging system that calculates the distance based on the known speed of light. The second generation of Kinect's sensor belongs to a sub category called range gated imagers, which uses a built-in shutter in order to modulate the frequency of the emitted light and a gate. This gate is located in front of the CCD chip of the camera. This principle was invented by Antonio Medina in 1992 (see [19]and [18]).

The basic principle of such a range gate imager can be best explained by using the phase detector block diagram shown in figure 1.2. The IR emitter (1) and the shutter (8) before the actual CCD image sensor are synchronously modulated by the modulator (2) with a periodical wave form. Thus there is a separation between light (open shutter) and no light (closed shutter) images, according to their corresponding half of the frame time. Those digital images generated while the shutter is open are stored in buffer number 3 and those images of the other half period (closed shutter) in buffer number 4. On the next occurring frame the on and off circles are reversed. Thus the new images are in synchronization with the light source, but exhibit opposed phase. This images are once again temporarily saved in the corresponding buffers (3 & 4).

The energy collected from an arbitrary pixel of the image during frame exposure time is schematically depicted as 3A curvature, while energy collected from the same pixel during the next frame exposure time is illustrated as 4A in figure 1.2. The digital value for that pixel will be the integrated value of the shaded area 3A (for buffer 3) or 4A (for buffer 4) for a time period of one frame.

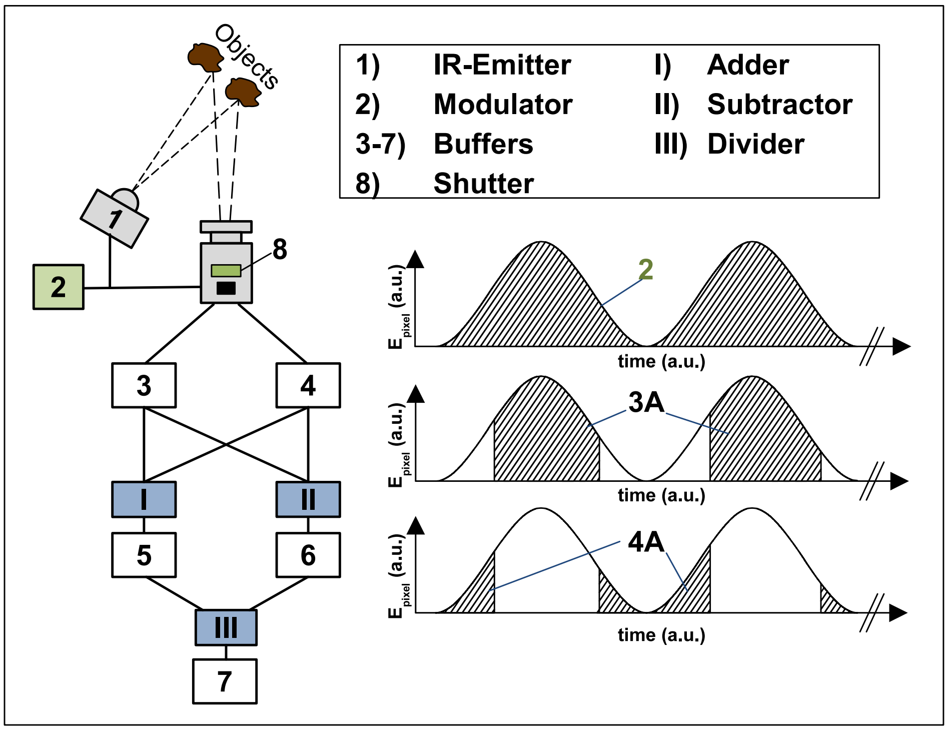


Figure 1.2: Common phase detector block diagram of a range gate imager using a sinusoidaly modulated illuminating beam of energy. (acquired from [18])

A pixel per pixel wise addition of buffer 3 and 4 leads to a conventional image of the scene, which is saved in buffer 5. On the other hand a subtraction of both buffers yield an image containing phase information of the scene. The phase image is saved in buffer nr. 6. The final resulting distances for each pixel can be obtained as a phase measurement by dividing relative phase (buffer 6) with the amplitude (buffer 5) of the pixel. In dependence of the distance to the corresponding picture element, the phase measurement will vary between the values -1 and + 1 for each pixel.

1.1.5. Data sources of the Kinect and their properties

The second version of the Kinect provides five main types of input data resources:

- The **audio stream** contains audio data at a 24-bit resolution captured by the microphone array (see figure 1.1). This data can be for example used to identify the direction of the audio sources, or for speech recognition.
- The **body stream** contains the overall position of the body and the 3D position of all available 25 joints in meters. These joint positions are the result of the internal skeleton tracking algorithm provided by the SDK. Up to six bodies are actively tracked.
- The **color stream** contains the image data captured by the RBG- camera of the Kinect sensor.
- The **depth stream** returns the current depth value of the scene for each pixel of a frame.
- The **infrared stream** allows to obtain a image of the scene without the influence of ambient light.

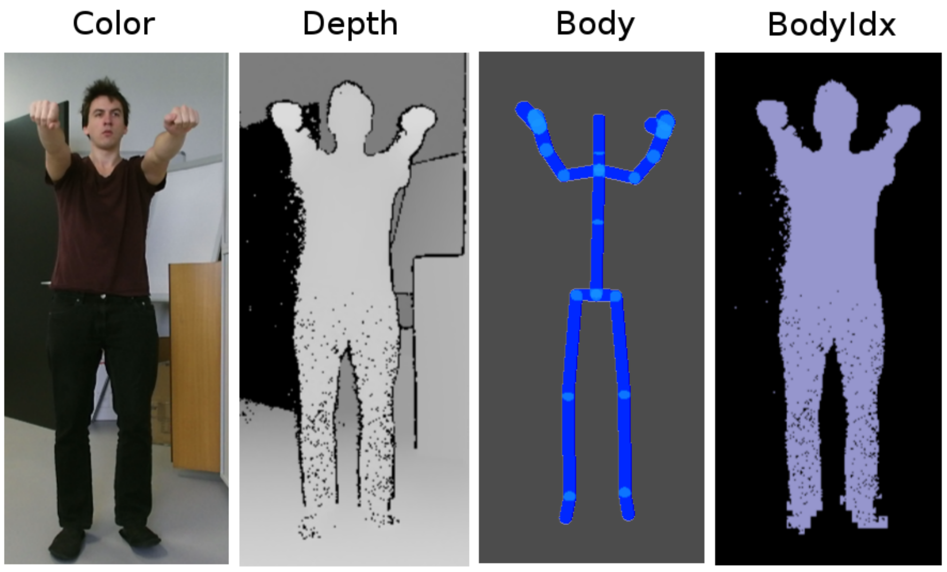


Figure 1.3: The used Kinect data sources.

Figure 1.3 illustrates the gained information of the input streams which were used for the SquatController software. On the left hand side there is the color stream aquired by the RGB-camera. The next sub-figure shows the depth stream of the same scene. Each pixel has a certain gray value which is proportional to the actual depth. The body stream provides a body object with a total number of 25 joints according to figure 1.4. For each joint the three dimensional positions (x,y,z) in meters are accessible. The developed SquatController software just needs the yellow and green marked joints (see figure 1.4) for a proper functionality. A connection of each joint yields the blue skeleton in figure 1.3. It demonstrates the visualization of the body stream of the same scene. With the combined information of depth-, infrared- and body stream, a segmentation between pixels belonging to a person and pixels belonging to the background is possible. This information is provided by the bodyIdx stream. The last sub figure on the right hand side illustrates this information gained by the bodyIdx stream for the same scene.

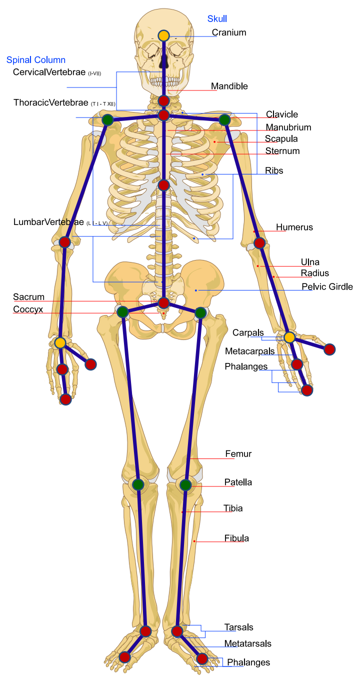


Figure 1.4: All 25 joints of the body object used by the Kinect v2. The green and yellow marked joints are used by the SquatController software.

1.2. Previous researches regarding the rehabilitation with the Kinect

The Kinect sensor was originally released exclusively for the Xbox console as a game controlling device. However with the release of the free Microsoft SDK in 2011 (see [21]) the door was opened for public usage. A huge variety of 3rd party applications in various categories emerged. The health care sector was one of them and has taken one of the highest research and development efforts. A substantial part of that sector uses the Kinect for rehab applications [17].

The Kinect sensor was not the first motion recognition technology used for monitoring rehabilitation exercises. However, most of the tracking devices rely on internal sensors or markers as tracking tools and thus the Kinect became a widely used and further investigated alternative. In many cases the basic tracking softwares were realized as game based approaches in order to increase the patient's motivation.

For example Lange et al. [7] developed and assessed an interactive game-based rehabilitation tool, with a focus on adults with neurological injuries. This tool was realized as a video game called ”JewelMine”, which was used in order to perform the balance training [14] [15].

Cervantes et al. [7] presented their work on cognitive and physical rehabilitation for Alzheimer patients using a Kinect-based video game.

A similar research using a Kinect based game for stroke rehabilitation was done by Saini et al. [27]. They focused on a game design principle with increased accuracy of stroke exercises targeting hand and leg rehabilitation, by evaluating the effect and feasibility of a new game technology.

Gotsis et al. [10] demonstrated a mixed reality game for upper body exercise using the Kinect sensor.

In order to facilitate the integration of full-body control with virtual reality applications using the Kinect, Suma et al. [29] introduced the ”Flexible Action and Articulated Skeleton Toolkit” (FAAST) .

Chang et al. [6] reports about a pilot study that assesses the possibility of rehabilitating two young adults with motor impairments using a Kinect. The obtained results according to the used ABAB sequence (A = baseline, B = intervention phase) showed that the two participants significantly increased their motivation for physical rehabilitation and thus the performance during the intervention phases, which uses the Kinerehab system.

Zhao et al. proposed a set of extensive and adaptable rules, in order to provide specific feedback, based on the violation of these rules [34]. In this context they further developed an at-home exercise monitoring system published in [33].

At the end of 2014 Microsoft released the second version of the Kinect sensor. Based on the revised sense technology called (time of flight) the new sensor exhibits an improved depth measurement accuracy. This upgraded sensor technology led to a variety of new research topics regarding the new Kinect sensor and its usage in the field of physiotherapeutic rehabilitation.

One of the first published investigations regarding the new sensor and the field of rehabilitation was done by Mottura et al. [24]. They presented a system called REAPP, which is a virtual environment aiming at supporting upper-limb robotic neurorehabilitation for post-stroke patients.

Ozturk et al. [26] evaluated the robustness and usability of the Microsoft Kinect in kinematic analyses of motor performance with stroke patients. Their primary objective was the identification of a kinematic metrics that best evaluates the impairments, by differentiating pathological performances between healthy subjects and stroke patients.

Mentiplay et al. [20] examined the concurrent validity and inter-day reliability of spatio-temporal and kinematic gait parameters. These parameters were estimated by using the Kinect v2 automated body tracking system and a criterion reference 3D motion analysis (3DMA) marker-based camera system. The spatio-temporal measurements had consistently excellent ($r \geq 0.75$) concurrent validity, with the exception of modest validity for medial-lateral pelvis sway($r = 0.45 - 0.46$) and fast paced gait speed variability ($r = 0.73$). In contrast kinematic validity was consistently poor to modest, with all associations between the systems weak($r = 0.50$). The conclusion exhibited that while the Kinect v2 body tracking may not accurately obtain lower body kinematic data, it showed great potential as a tool for measuring spatio-temporal aspects of the gait.

A design research of an interactive monitoring tool, using the Kinect v2, which allows subjects to carry out tailored exercises for home-based physical rehabilitation was performed by Capecci et al [5]. While the algorithm is able to compare these features with those computed by a reference subject, the available systems were yet unable to satisfy specific defined clinical and technical requisites.

Springer et al. [28] critically evaluated the literature describing the concurrent validity of using the Kinect as a gait analysis instrument. Their search identified 366 papers, from which 12 relevant studies were retrieved. As a result the measurement of kinematic parameters showed poor validity and large errors. But they concluded that the Kinect may have the potential to be used as a tool for measuring spatio-temporal aspects of the gait, yet standardized methods should be established, and future examinations with both healthy subjects and clinical participants are required in order to integrate the Kinect as a clinical gait analysis tool.

1.3. A squat or knee bend from the therapeutic viewpoint

In order to create a rehab training tool, capable of tracking a human's knees and furthermore verifying whether the the performed knee bend is correct, the specific knowledge of a therapeutically correct knee bend is necessary. With the help of two professional physiotherapists at the rehab clinic Judendorf Straßengel, the underlying conditions of such a knee bend was elaborated.

The initial posture is given by placing the legs hip-width apart and straightening the shoulders and the back, indicated by the blue line in figure 1.5.

While performing a knee bend, it is of great importance, that the knees do not overtower the toe tips of the same leg and that the back remain in a straight manner. The red line in figure 1.5 schematically shows the boundary behind which the knees should remain. Further attention should be paid in respect to unwelcome knee rotations while performing a knee bend. The knees should remain in a straight line from a frontal viewpoint (see figure 1.5 right side). Reason for both regulations are a decrease of the burden of the knees and an uniform distribution of the load on the thighs and the buttocks.

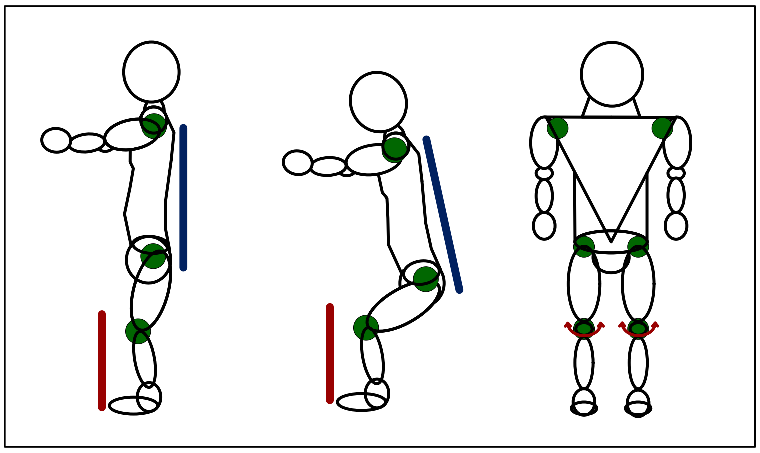


Figure 1.5: Scheme of a correct knee bend from a therapeutic viewpoint

1.4. Kinect4Rehab as a vision

In chapter 1.2 plenty of researches regarding the field of tele-rehabilitation with the Microsoft Kinect sensor were presented. The basic idea of all mentioned investigations was to develop a software solution in order to allow the tracking of specific physiotherapeutic exercises.

This thesis was intended to develop a software solution for a rehab training tool in order to track a knee bend or squat exercise. By adopting the fundamental structure of the controlling routine of the SquatController, the construction of a knee bend tracking module should be realizable with little effort. The creation of this module would be one of the first steps in order to be part of a much greater project, the Kinect4Rehab project.

The concept of this newly considered project lies in a software that enables the possibility of tracking more than just one specific exercise or serving one specific purpose. A variety of different exercises should be includable by just loading extern exercise modules.

2. Software description of the Squat Controller software

This chapter gives a detailed description of the developed SquatController software and is separated in three chapters. It was developed in C# with Microsoft Visual Studio as a WPF-project.

The first subchapter 2.1 provide an overview as well as detailed information about the operating principle of the SquatController software. The next subchapter 2.2 explains the fundamental algorithms that are essential for an accurate tracking of the knees and the calculation of the provided feedback. Sub chapter 2.3 describes the basic concept of the used design pattern of the SquatController software. This Model-View-ViewModel pattern introduced by Gossman [9] allows an additional level of encapsulation between Model, View and ViewModel.

2.1. The operating principle

Probably the best way for describing the operating principle of the software is to split it into two fundamental parts. The main section is handling the input and output of the program and the operating section which processes identifiers and parameters. This particular viewpoint can be linked to the two accordingly used threads of the software, the main thread and the background working thead.

Figure 2.1 illustrates the data exchange between main and background working thread in a simplified way. The initial entry of specific input parameters (i.e. SHANK_LONG_DIAMETER, FOOT_SIZE, etc...) via the GUI is necessary in order to allow the background working thread to define the corresponding boundaries on which the subsequent feedback is based. The main thread functions as the manager for in- and output data. It acquires the input parameters from the GUI (display) and the raw source data from the Kinect and forwards these data to the background working thread. The background working thread processes the current data and calculates four feedback parameters depending on the current state of the knee bend. The information of these four parameters as well as the video stream of the RGB - camera of the Kinect (color stream) will thereafter be used by the main thread in order to visualize it on the feedback display of the GUI.

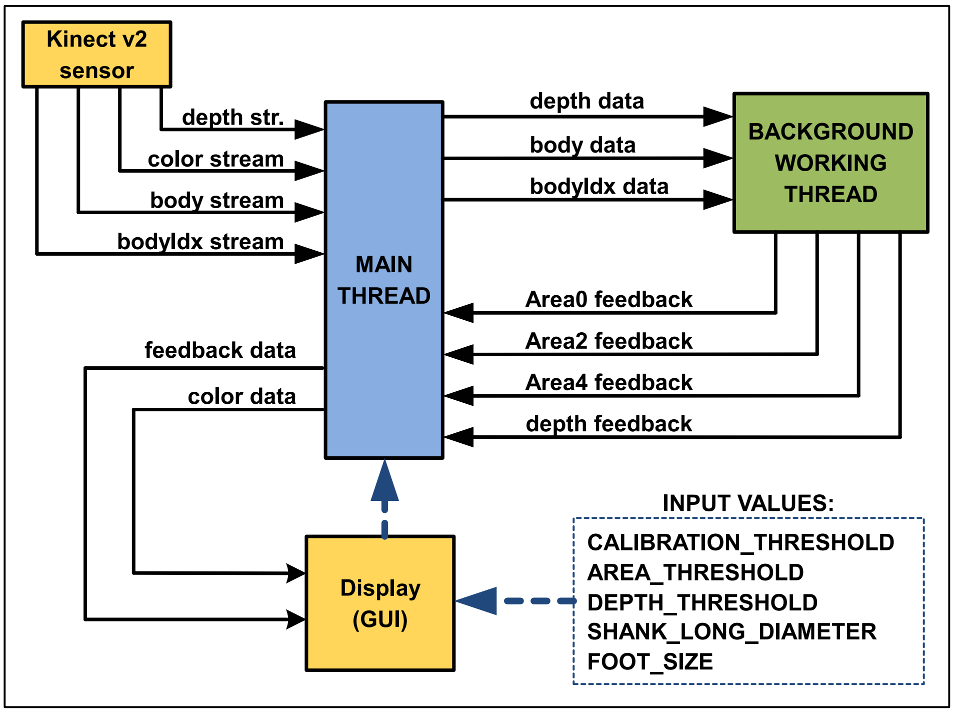


Figure 2.1: Simplified scheme of the main and background working thread of the Squat-Controller and its data exchange.

2.1.1. The main thread

The first step the main thread takes, is to create a MainWindowViewModel object. This particular class handles a variety of different tasks crucial for the functionality of the SquatController software. Besides of managing properties which are needed for the GUI, it is directly connected with the Kinect sensor. It receives four data streams which are converted to usable data for further utilization. During initialization an InBdrController object is defined. This object is necessary for starting the background working thread by invoking the CheckSubejct function (described in more detail in chapter 2.1.2).

Figure 2.2 gives an overview of the most crucial functions of the MainWindowViewModel-object. The blue marked fields are public methods belonging to the InBdryController object which are frequently invoked by the main thread in order to update frame related data. At first the initialization of the members, event handlers and the view related properties of the object and the initialization of the background working thread are performed. Thereafter the main thread waits for incoming data streams, provided by the Kinect sensor, which are triggered by the events of the MultiSourceFrameReader.

The MultiSource object provided by SDK of Microsoft is a combined data resource including all before mentioned resources (see chapter 1.1.5). The big advantage of using the

MultiSource instead of the single resources lies in synchronization between the requested frames. In case of the SquatController software, the four streams which can be seen in figure 1.3 were requested. Each of them are further processed by their corresponding Show...Frame() functions:

- **ShowBodyIdxFrame** checks whether a bodyIdx frame is available, and if it is the case it calls the ProcessBodyIndexFrameData method. This method is used for separating background data from subject specific data. The particular information which pixel belongs to a subject and which pixel does not, is used for the tracking routine of the SquatController itself and thus handed over to the background working thread by calling the updateBodyIdxData-method of the InBdrController - object.
- **ShowBodyFrame** checks whether a body frame is available, and if it is the case, calls the updateColorDrawingContext method. This method is used for manipulating the color stream in order to visualize the locations of the used joints as colored points. From the original 25 joints provided by Microsoft's SDK, only 6 were used. (see figure 1.4, green and yellow marked circles). By using the updateJointData method of the InBdrController object, an update on the current locations of each of the joints and each frame is made to allow the backround working thread to work with up-to-date data.
- **ShowDepthFrame** checks whether a depth frame is available, and if it is the case it calls the ProcessDepthFrameData method, which obtains a depth value for each pixel and each frame of the depth stream and assigns it to the corresponding matrix entry. Those particular depths are the distances between the Kinect sensor and reflection points of the IR-ray. The resulting matrices are made available for the InBdryController - object by invoking the updateDepthData -method.
- **ShowColorFrame** checks whether a color frame is available, and if it is the case, converts each pixel and each frame of the obtained stream data into the corresponding color pixel of the used visualization. It is embedded as WritableBitmap object in the graphic user interface. Additionally it calls the updateFeedbackGraph - method, which updates the feedback visualization in dependence of the four feedback parameters.

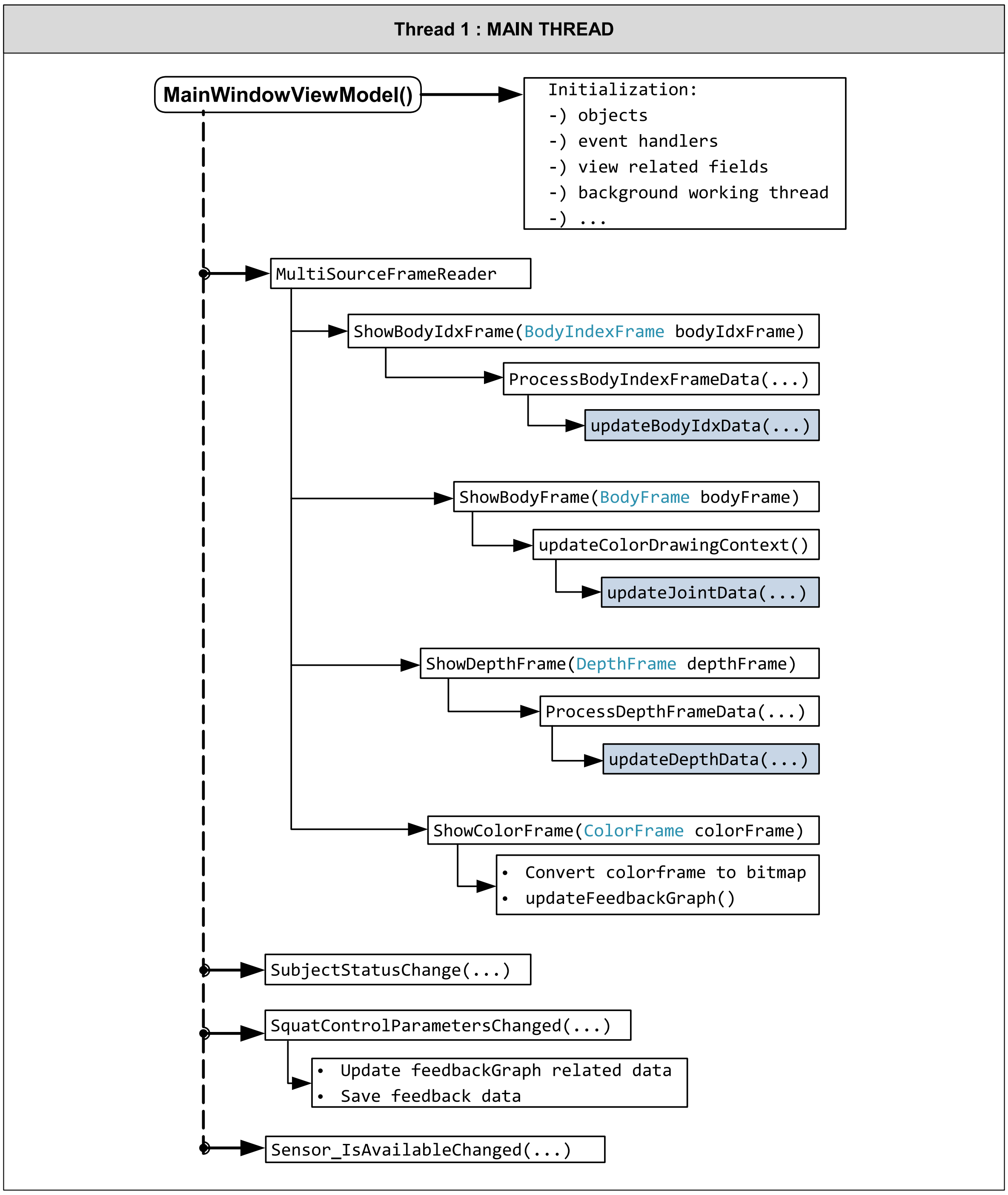

Thread 1 : MAIN THREAD
MainWindowViewModel()
Initialization:
-) objects
-) event handlers
-) view related fields
-) background working thread
-) ...
MultiSourceFrameReader
ShowBodyIdxFrame(BodyIndexFrame bodyIdxFrame)
ProcessBodyIndexFrameData(...)
updateBodyIdxData(...)
ShowBodyFrame(BodyFrame bodyFrame)
updateColorDrawingContext()
updateJointData(...)
ShowDepthFrame(DepthFrame depthFrame)
ProcessDepthFrameData(...)
updateDepthData(...)
ShowColorFrame(ColorFrame colorFrame)
• Convert colorframe to bitmap
• updateFeedbackGraph()
SubjectStatusChange(...)
SquatControlParametersChanged(...)
• Update feedbackGraph related data
• Save feedback data
Sensor_IsAvailableChanged(...)

Figure 2.2: Simplified flowchart of the main thread of the SquatController software, realized in the `MainWindowViewModel.cs` file.

Next to the MultiSourceFrameReader three additional events are necessary to mention.

If a change of the subject's position is detected during exercise performance, the Subject-StatusChange event will be triggered by a triggerpoint generated by the InBdrController object. The MainWindowViewModel object responds to this triggerpoint and therefore knows when an update of the calibration status display is necessary.

The SquatControlParametersChanged event updates the feedback parameter entries used for the visualization of the current state of the squat. Each time new parameters are available, it will be triggered by the OnSquatControlParametersChanged method of the InBdryController object. Thus an update of the corresponding parameters of the main thread is initiated. The event can additionally be used as trigger method for saving the current parameter values. Therefore only new occurring values are saved by calling the PushResultsOnLogger method.

The Sensor_IsAvailableChanged event provided by the SDK of Microsoft indicates whether the Kinect sensor is connected properly or not. By using this event there is a simple way to supply the user with context information via the graphic user interface in case of such obstacles.

2.1.2. Background working thread

The main task of the background working thread is to process identifiers and parameters from the data which are provided by the main thread. While the thread itself is initialized in the MainWindwoViewModel object, all corresponding methods are summarized in the InBdryController object. In order to start the background working thread, the CheckSubject method has to be invoked which forces the thread to stay in an infinite loop, as long as the main thread is running. Figure 2.3 shows a simplified flowchart of SquatController's background working thread.

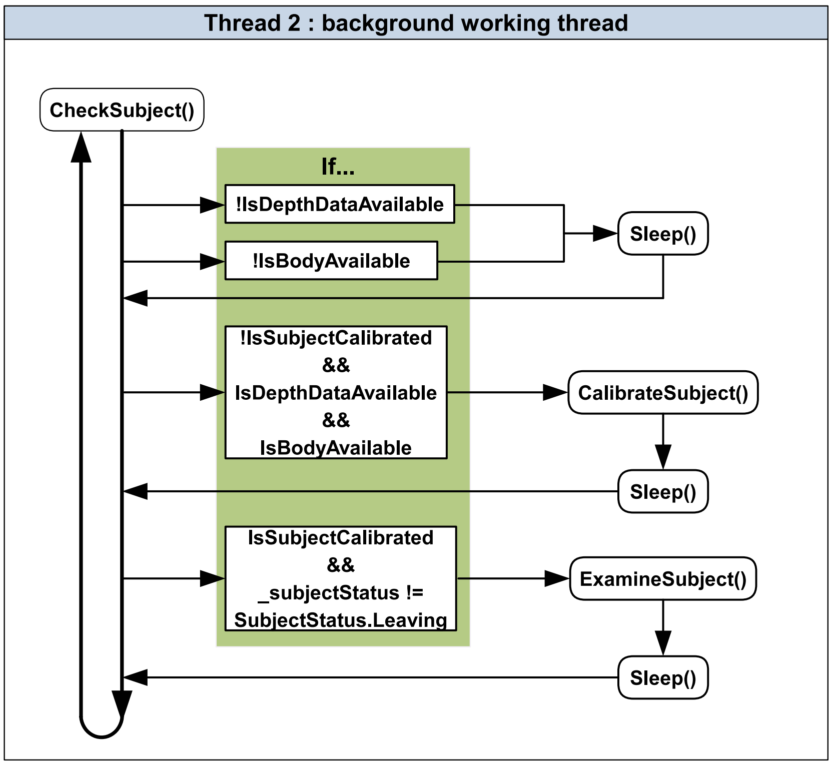

Thread 2 : background working thread
CheckSubject()
If...
!IsDepthDataAvailable
!IsBodyAvailable
Sleep()
!IsSubjectCalibrated && IsDepthDataAvailable && IsBodyAvailable
CalibrateSubject()
Sleep()
IsSubjectCalibrated && _subjectStatus != SubjectStatus.Leaving
ExamineSubject()
Sleep()

Figure 2.3: Simplified flowchart of the background working of the SquatController, realized in the `InBdryController.cs` file.

As long as no body can be detected and no depth related data, provided by the main thread, is available, it repeatedly enters a sleep period, which lasts for about 2 seconds for each loop cycle. If both states are true, meaning a body is facing the sensor and depth data is available, the program enters a calibration mode by calling the CalibrateSubject method.

In Figure 2.4 the principle of the calibrateSubject() method is illustrated. The first step of this particular function is that it enters an infinite loop. While within the loop the program determines if the subject is holding a specific defined calibration position (described in chapter 3.3). To do so, it is comparing the distances between wrist and head for both arms (see yellow marked joints in figure 1.4 in chapter 1.1.5) at two different time points. If the distances of both time points are approximately equal and within a certain threshold, the current position is accepted as a calibration position. If this is the case, the "IsSubjectInCalibrationPosition" flag is set and program specific reference values. Those reference values are often used during subject observation and therefore used as member variables of the InBdryController object.

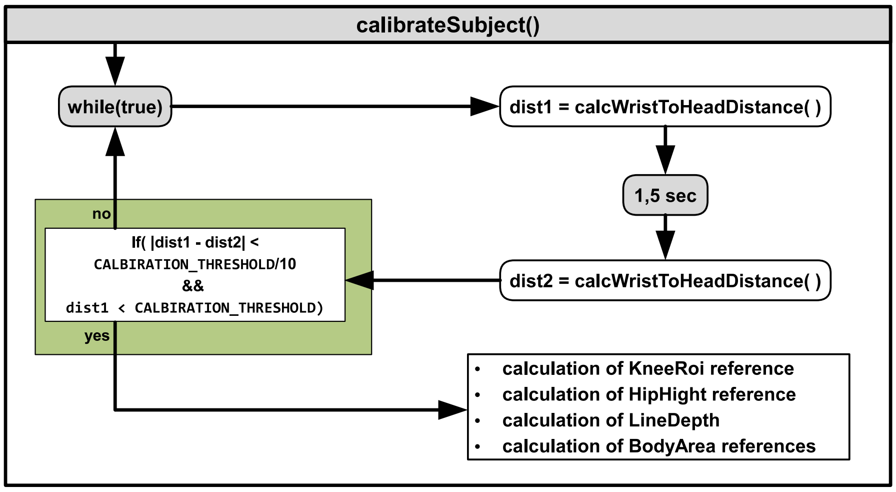

calibrateSubject()
while(true)
dist1 = calcWristToHeadDistance( )
1,5 sec
dist2 = calcWristToHeadDistance( )
no
If(|dist1 - dist2| <
CALBIRATION_THRESHOLD/10
&&
dist1 < CALBIRATION_THRESHOLD)
yes
calculation of KneeRoi reference
calculation of HipHight reference
calculation of LineDepth
calculation of BodyArea references

Figure 2.4: Simplified flowchart of the calibrateSubject method of the InBdryController object.

The ExamineSubject method (see figure 2.5) is called under the condition that the IsSubjectCalibrated flag is true, which means that a calibration already was done. Initially the subject's position is once again checked by calling the CheckIfInInitialPosition method. If there was a change in position a new calibration is forced automatically by setting the IsSubjectCalibrated flag to false and thus the CalibrateSubject method will be automatically entered at the next loop cycle. Elsewise a current region of interest (KneeROI) of the particular frame is determined. This KneeRoi object can be seen as a rectangular moving window targeting the center of the participant's knees and thus indicating the relevant data area for further utilization .

The calculateStartEndIdx method is used to determine the exact indices of start and endpoint of the KneeRoi. These indices are used to limit the loop cycles used for pixel to pixel observations of depth and bodyIdx data . Within the mentioned ROI, the pixels are

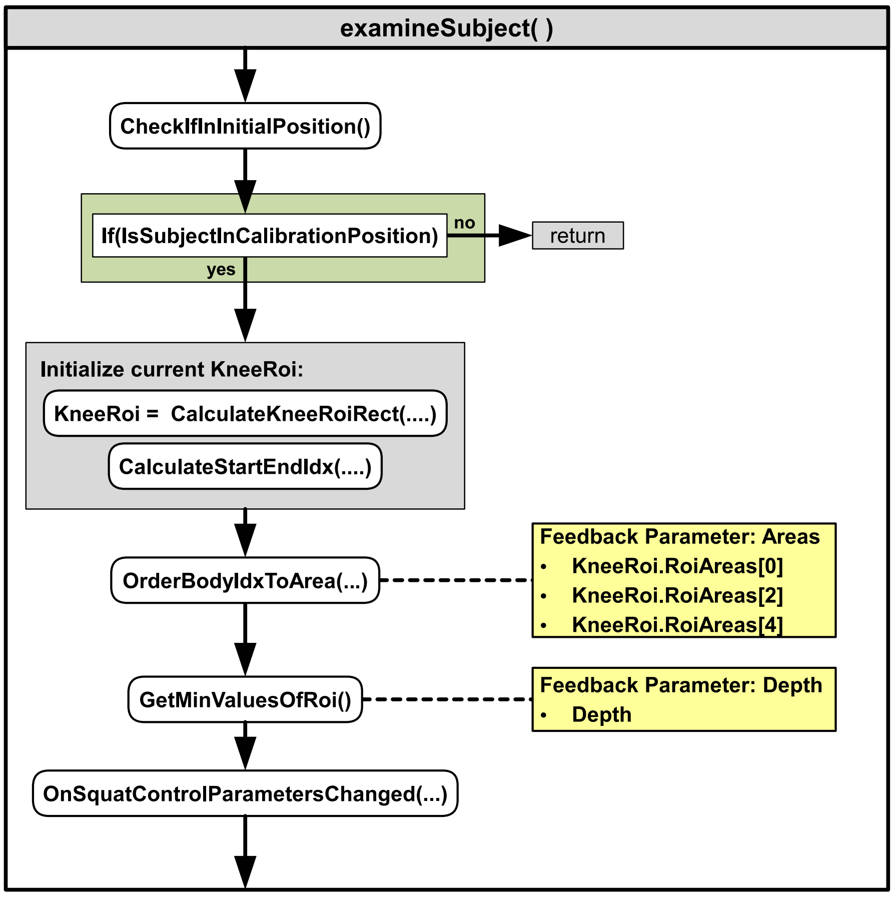

examineSubject( )
CheckIfInInitialPosition()
If(IsSubjectInCalibrationPosition)
no
return
yes
Initialize current KneeRoi:
KneeRoi = CalculateKneeRoiRect(....)
CalculateStartEndIdx(....)
OrderBodyIdxToArea(...)
Feedback Parameter: Areas
KneeRoi.RoiAreas[0]
KneeRoi.RoiAreas[2]
KneeRoi.RoiAreas[4]
GetMinValuesOfRoi()
Feedback Parameter: Depth
Depth
OnSquatControlParametersChanged(...)

Figure 2.5: Simplified flowchart of the examineSubject method of the InBdryController object and the locations where the feedback parameters are calculated (marked yellow).

separated in five areas depending on whether the pixels belong to the subject or to the background. This separation occurs within the OrderBodyIdxToArea function which is further explained in chapter 2.2.1. Three out of the five areas, those who are not belonging to the subject, are used as feedback parameter and therefore saved for later usage.

The GetMinValuesOfRoi method returns the minimum depth value of the KneeRoi by direct comparison of each pixel. With the additional information from the bodyIdx data, the amount of the observed pixel can be further reduced to those pixels which are belonging to the subject within the ROI and therefore a further decrease of processing power is accomplished. The result of the method is another feedback parameter, which is used for the online feedback visualization.

By calling the OnSquatControlParametersChanged method, a new event trigger point, including the actual information of all four feedback parameters, is generated. This trigger point forces the call of the SquatControlParameterChanged event by the main thread and therefore an update of the new feedback parameters by the main thread is initiated.

2.2. Key sections in detail

This section includes a summary as well as a detailed description of the essential algorithms of the SquatController software.

2.2.1. OrderBodyIdxToArea method

The OrderBodyIdxToArea method separates pixels within the KneeROI in 5 subareas depending on whether the pixels belong to the body or to the background. The sum over all pixels of each subarea is further used as a feedback parameter for the visualization.

Figure 2.6 illustrates an example of such a KneeROI. A for-loop was used to iterate through all rows of the matrix and sort it into 5 regions depending on the following rules:

1. Area0: All pixels within a row from start to first body pixel
2. Area1: All pixels within a row from first body pixel until the first subsequent background pixel
3. Area2: All pixels within a row from the end of Area1 until the first subsequent body pixel.
4. Area3: All pixels within a row from the end of Area2 until the first subsequent background pixel
5. Area4: All pixels within a row from the end of Area3 until the end of the row.

Pixels belonging to the body are highlighted green in figure 2.6. With the information gained from the bodyIdx resource a simple distinction between pixels belonging to the body and pixels belonging to the background is possible.

In order to decrease the error, which is caused by uncertain area assignments, two similar but independent sorting paths were used. One path (orange), sorts all pixel from the left to the right side and from top to bottom and the other path (blue), sorts all pixel form the right to left side and from bottom to the top. The average of both independently acquired values for each of the areas are then used as the actual area parameter. As feedback parameter for the visualization serve Area0, Area2 and Area4.

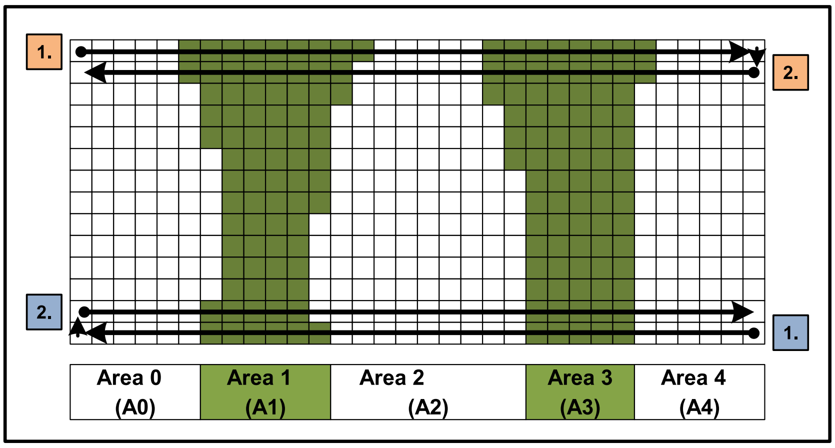


Figure 2.6: Scheme of a KneeRoi with background pixels (white) and body pixels (green).

2.2.2. GetMinValuesOfRoi method

The GetMinValuesOfRoi method searches for the minimum values of the right and left half of the KneeROI in order to obtain the depth of the right and left knee. Therefore a for-loop was used to iterate through each pixel of the whole ROI. If a depth value of the current pixel is smaller than the current minimum depth value, the new depth value will be used as new depth minimum. Thus an iteration through all pixels of the ROI results in the determination of both global minima, one from the left side and one from the right side. By averaging over both minima the depth parameter for the particular frame is calculated.

2.2.3. UpdateFeedbackGraph method

The updateFeedbackGraph method updates the feedback visualization for each new color frame according to the four feedback parameters, Depth, Area0, Area2 and Area4. These parameters are continually updated by the SquatConrtrolParameterChanged event.

Figure 2.7 shows the feedback visualization as it is realized in the GUI of the SquatController. Additionally the location of the reference values as well as the limits for depth and area parameters are labeled in the figure. Area0 and Area2 have the same reference and limit values as Area4 with regard to their specific axes. The occurring feedback values during the measurement are then visualized in proportion to these labeled reference and limit values.

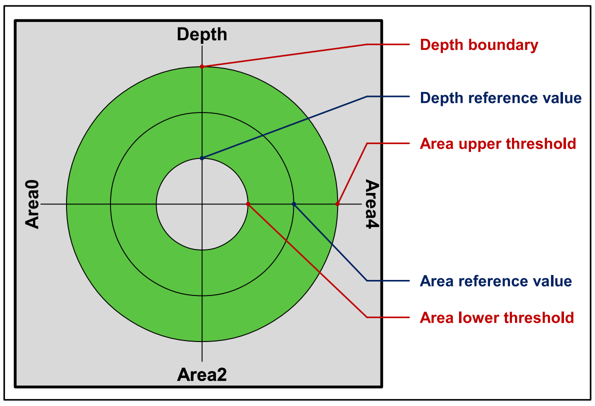


Figure 2.7: Feedback display of the Squat Controller and its configurable boundaries.

The reference values for each of the parameters are obtained during the calibration mode. The calculation of the depth boundary value was performed as indicated in equation 2.1. It describes the distance between sensor and toe tips, which embodies the boundary which must not be exceeded in order to perform a correct squat (see chapter 1.3).

$$Depth\ boundary = depth\ reference\ value - \left(FOOT\_SIZE - \left(\frac{SHANK\_LONG\_DIAMETER}{2}\right)\right) \tag{2.1}$$

The limits for each lower and upper threshold regarding the area parameters are calculated as described in 2.2.

$$Area\ threshold_i = area\ refrence_i \pm AREA\_THRESHOLD \tag{2.2}$$

$FOOT\_SIZE$, $SHANK\_LONG\_DIAMETER$ and $AREA\_THRESHOLD$ are input parameters obtained from the graphic user interface and therefore an initial configuration for each subject is necessary.

An additional moving average filter was implemented in order to allow a smoother response of the feedback visualization. This filter was applied on each of the four feedback parameters, but while 15 samples were used for the depth parameter the usage of 8 samples for the area parameters were sufficient.

Only after a successful initial calibration the visualization of the current feedback values in proportion to their relating reference and limit factors is possible. For the illustration of the feedback, connecting lines between the feedback values are drawn to make it easily comprehensible. These lines result in a certain geometric form. The best case, i.e a perfect squat, would be a rhombus. While within the green highlighted area the knee bends are considered as correct, like it is shown in figure 2.8a.

According to chapter 1.3 there are two important rules to consider if someone intends to perform a proper physiotherapeutic knee bend. The first rule implies that the knees must not overtower the corresponding toe tips of the same leg. The second rule suggests to avoid unwelcome knee rotations while performing a knee bend. According to this two rules three error cases were formulated:

1. **exceeded frontal boundary** - If the knees exceeded the depth boundary (= depth of toe tips), figure 2.8b
2. **knees turned inside** - If the knees turned inside and exceeded a certain area threshold, figure 2.8c
3. **knees turned outside**- If the knees turned outside and exceeded a certain area threshold, figure 2.8d

For each of these three incorrect cases of the knee bend, the geometrical form of the feedback display shows a certain distortion (see figure 2.8b-c).

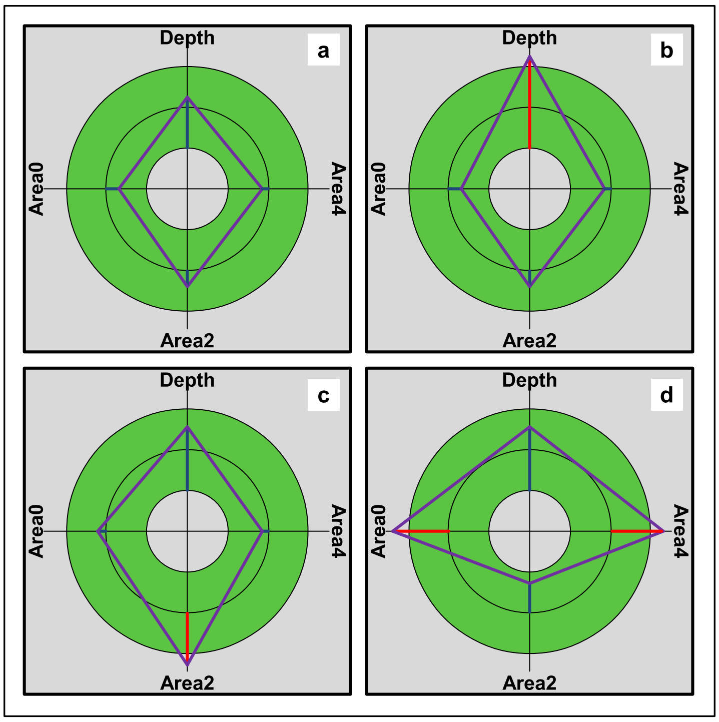


Figure 2.8: Various feedback display configurations: a) correct knee bend, b) exceeded frontal boundary error, c) knees turned inside error, d) knees turned outside error.

2.2.4. CheckIfInInitialPosition method

This particular function checks if the subject has changed the position during a running measurement.

At the very beginning it determines if the subject is in standing position. If this is the case it calculates the average depth of the subject's left and right side in relation to the used joints (see figure 1.4 in chapter 1.1.5, green marked joints), as it is described in equation 2.3 and 2.4.

$$\overline{RightSideDepth} = \frac{(ShoulderRight_Z + KneeRight_Z + HipRight_Z) * 1000}{3} \tag{2.3}$$

$$\overline{LeftSideDepth} = \frac{(ShoulderLeft_Z + KneeLeft_Z + HipLeft_Z) * 1000}{3} \tag{2.4}$$

If both depth values differ more than a certain DEPTH_THRESHOLD, the program assumes that the subject's position has changed and is not facing the sensor in a perpenticular manner. Therefore a new calibration will be initiated.

Additionally the average depth of all six joints is calculated by calling the getRefDepth method. If there is more than a certain DEPTH_THRESHOLD between the current calculated mean depth and the initially recorded reference depth, a new calibration will be initiated.

The so called DEPTH_THRESHOLD can be adjusted via the graphic user interface.

2.3. Model-View-ViewModel-Pattern

In this chapter the basic concept of the used Model-View-ViewModel pattern is described.

Figure 2.9 shows a simplified scheme of such a MVVM design structure, including some examples of the used classes of the SquatController, sorted into their belonging categories. It basically consist of a Model, a ViewModel and a View which resembles the GUI. A short summary of the basic principles of MVVM will follow, but for a more detailed description about the MVVM pattern please refer to [23].

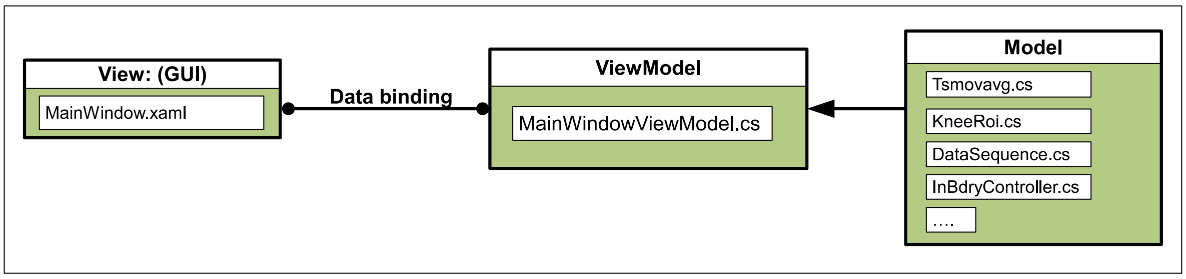


Figure 2.9: Simplified scheme of SquatController's MVVM structure.

2.3.1. Model

The model can be seen as a synonym for data like entries, classes or databases. Classes like `InBdrController.cs`, `KneeRoi.cs` and many more, are typical examples of the model section of the SquatController. Within the model are the data behind the scene.

2.3.2. View

The view section concerns the graphic user interface (GUI). It was written in XAML syntax, a declarative XML-based language. All significant parameters are provided by the ViewModel via parameter binding. This binding method connects a certain GUI element directly to a specific property of the ViewModel. If this property changes during the execution of the program, the corresponding GUI element will alter in the same way. Therefore a later change of the current design is possible without undertaking great efforts in adjusting the original program source. The only necessary step is the new connection between XAML objects and the specific properties of the ViewModel.

2.3.3. ViewModel

The ViewModel of the SquatController software operates as the manager for data in- and output. It is directly connected with the Kinect sensor and receives the data streams from the Kinect. These resource streams are then converted into usable data for further utilization. Additionally the ViewModel operates as a manager for view-specific data. If there are some parameter modifications during the execution of the program (i.e depth feedback), the corresponding XAML object will automatically be updated due to the data binding.

3. Test measurement

In order to gain initial insights, whether the SquatController software can be used as rehab based assisting tool, some test measurements were performed. This chapter gives detailed information about the used devices and explains the used measuring setup and procedure of these test measurements. Furthermore it includes the subject specifications and a description of the performed test evaluations.

3.1. Used Devices and Equipment

During the measurement, the following devices were used:

- Kinect for Windows V2 Developer Bundle (sensor + cable)
- Laptop with the necessary hardware requirements (detailed information in chapter A.2)
- Tripod and video projector
- Kinect for Windows SDK 2.0 + drivers
- SquatController

The off-line analysis and calculations were performed in Matlab. For a more detailed information about the used program versions please refer to the appendix chapter A.1.

3.2. Measurement setup

The recording was done with the second version of Kinect's sensor, which was connected via USB v.3 cable with a PC equipped with the proper hard- and software requirements. The sensor was mounted on a tripod at a height of one meter above ground level. The subjects took position in front of the sensor at a distance of about two and a half meters. A video beamer was used to extend the PC's display to provide an apparent feedback for the subject.

3.3. Measuring procedure

Every subject underwent three measurement sessions. The only difference between these three sessions was the AREA_THRESHOLD parameter, which defines the boundaries of a correct knee bend regarding the area parameters of the feedback visualization. Thus an AREA_THRESHOLD of 150, 175 & 200 was tested. The variation aims on finding a

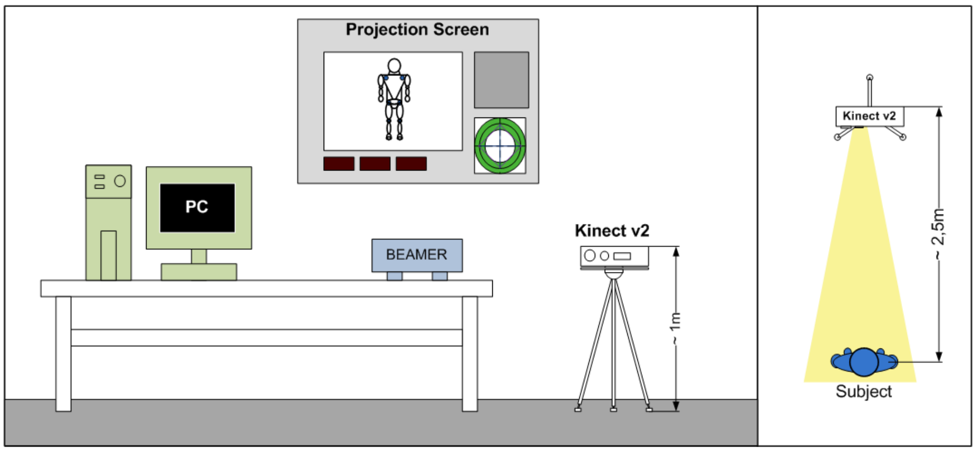


Figure 3.1: Scheme of measurement setup.

suitable value for all subjects. In order to do so each exceedance of each area feedback parameter (Area0, Area2 & Area4) was documented.

Before the measurements started three anatomical parameters were determined, the leg length from hip joint to the ankle, the shank long diameter at ankle height and the feet size. Shank long diameter and feet size are necessary input parameters which can be entered via the GUI of the SquatController software.

Each of the three measurements started with the initial calibration. To do so the subject had to take position in front of the displayed red line and put both palms behind the head (see figure 3.2, top). It is necessary to stand as still as possible for about two seconds until the calibration routine is finished. If the calibration is successful, the corresponding indicator light changes the color from yellow to green.

After a successful calibration, the subject was asked to perform 20 knee bends. A knee bend in this particular experiment was defined as a continuous motion between upright standing position and knee bend position (see figures 3.2, bottom) within a time period of 1,5 to 2 seconds. Additionally, instructions on how to perform a proper knee bend from the psychotherapeutic viewpoint (see chapter 1.3), were provided.

During the measurement the subjects were provided with a visual feedback via the GUI. The main task was to remain within the boundaries (green area in figure 2.8) while performing the exercises.

3.4. Test subjects

Data were recorded from six healthy subjects. One half was female, the other half male. All subjects were in the range of 26 $\pm$ 2 years. The ascertained anatomical parameters

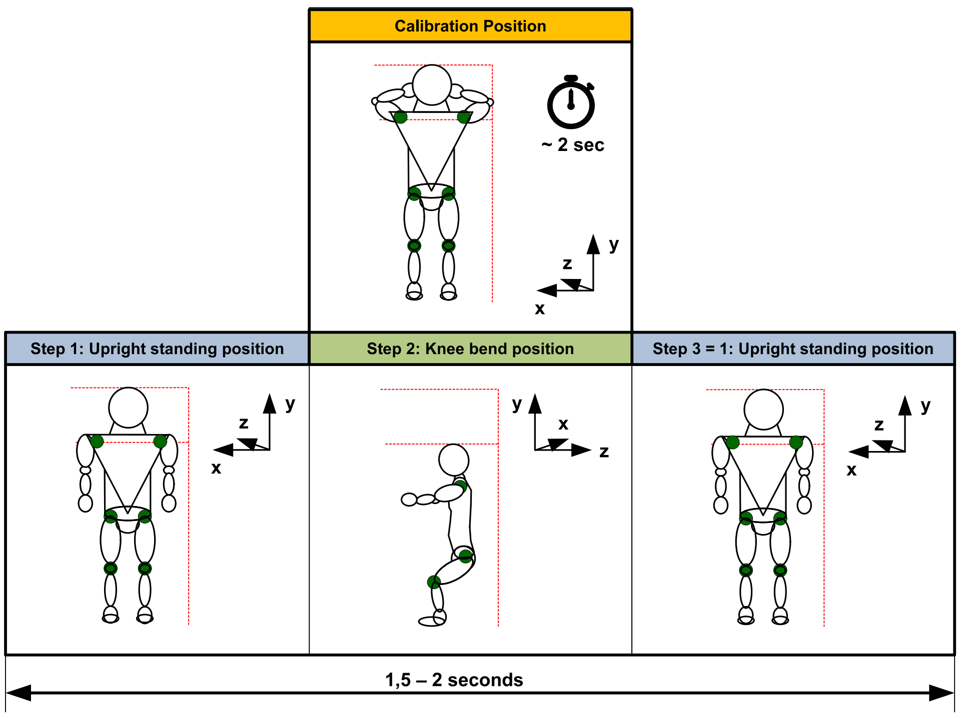

Calibration Position
~ 2 sec
y
z
x
Step 1: Upright standing position
Step 2: Knee bend position
Step 3 = 1: Upright standing position
1,5 – 2 seconds

Figure 3.2: Scheme of measuring procedure, calibration position (top) & knee bend (bottom)

for each of the subjects are summarized in table 3.1 in the results section.

3.5. Software evaluation

Two different approaches were made for evaluating the SquatController software.

3.5.1. Usability test

Each subject was asked to undergo a test consisting of 10 questions regarding the software's usability. This particular usability test was obtained from [4]. A copy of the original used file is attached at the appendix in section A.3.

The evaluation itself was done by calculating a certain usability factor which yields a single composite measure of the overall usability of the system being studied. For the calculation of the usability factor it is necessary to sum the score contributions. Odd numbered questions have a score range from 0 to 4 points, whereas the score range of even numbered questions starts at 4 and proceeds to 0 points.

By summarizing all points from each question and multiplying it by 2.5, a total score can be obtained which ranges from 0 to 100. The higher the value is, the higher is the usability of the software.

3.5.2. Evaluation of recorded depth data

A test which compares the feedback depth values, calculated by the GetMinValuesOfRoi method (see chapter 2.2.2) and depth values obtained by an additional offline analysis was performed. In order to do so, raw depth data limited by the KneeRoi was saved in external .DAT files. With the help of a self scripted Matlab function called `Import_SQC_Data.m` it is possible to import data from those files into the workspace for further calculations. The actual depth evaluations are performed with another scripted Matlab file named `Data_Evaluation.m`.

Each frame of the depth data was preprocessed by a median filter with a 3x3 structure element. Afterwards the global minimum for each frame was calculated. These minima are used as reference values for estimating the deviation between reference and depth values used as feedback by the SquatController software.

Equation 3.1 describes the estimation of the reference depth values ($ref\_depth_{i,offline}$) for each frame i, calculated from the raw depth data of the offline analysis.

$$ref\_depth_{\,i,offline} = min\left(median_{\,3\times3}\left(depth\_data_{\,i,raw}\right)\right) \tag{3.1}$$

The deviation, labeled as "error", between reference depth value and used feedback depth value for each frame, was calculated as described in equation 3.2.

$$error_i \;= ref\_depth_{\,i,offline} - feedback\_depth_{\,i} \tag{3.2}$$

For each of the 6 subjects the mean, the 10^{th} and the 90^{th} percentile of the deviation were calculated. 10^{th} and 90^{th} percentile can be used to observe the behaviour of 80% of the data by examining the differences (see eq. 3.3).

$$dist_{80\%} = prct_{90}(error) - prct_{10}(error) \tag{3.3}$$

3.6. Results

This chapter summarizes the observed results in respect to the initially performed anatomical measurements, the results of the documented amount of correct knee bends, the realized usability tests and the results in respect to the performed depth analysis.

3.6.1. Obtained anatomical parameters

In order to provide the software with the necessary anatomical input parameters the leg length, from hip joint to the ankle, the shank long diameter at ankle hight and the feet size were measured. The results of those measurements are listed in table 3.1.

Table 3.1: Subject specific anatomical parameters.

subjects	gender	leg length	shank diameter (long)	feet size
		mm	mm	mm
subj01	w	830	90	250
subj02	w	900	100	260
subj03	w	840	80	240
subj04	m	900	100	260
subj05	m	890	100	280
subj06	m	890	90	230

3.6.2. Results regarding the AREA_THRESHOLD parameter

The summarized outcome of the observed amount of correct performed knee bends for each measurement and each subject are listed in table 3.2. Additionally the mean over all subjects for each area feedback parameter and each of the three AREA_THRESHOLD values, were calculated in order to allow a quicker comprehension.

Table 3.2: Summarized results regarding the AREA_THRESHOLD parameter for each subject.

subjects	AREA_THRESHOLD: 150			AREA_THRESHOLD: 175			AREA_THRESHOLD: 200		
	Area0 %	Area2 %	Area4 %	Area0 %	Area2 %	Area4 %	Area0 %	Area2 %	Area4 %
subj01	100.0	0.0	100.0	100.0	65.0	95.0	100.0	55.0	100.0
subj02	100.0	90.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
subj03	95.0	25.0	100.0	100.0	55.0	100.0	100.0	55.0	100.0
subj04	95.0	100.0	100.0	100.0	80.0	100.0	100.0	95.0	95.0
subj05	100.0	25.0	95.0	100.0	80.0	95.0	100.0	85.0	100.0
subj06	100.0	45.0	90.0	100.0	95.0	100.0	100.0	100.0	100.0
mean(subj)	98.3	47.5	97.5	100.0	79.2	98.3	100.0	81.7	99.2

3.6.3. Results of usability evaluation

Table 3.3 summarizes the results according to the performed usability test. For each question a score between 0 and 4 points can be achieved. The higher the score the better is the usability. The last column of the table shows the total score for each subject. It ranges from 0 to 100 point, 100 points being the maximal achievable score.

Table 3.3: Evaluated score results of the usability tests for each subject. Score ranges from 0 (worst) to 4 (best), and total score ranges from 0(worst) to 100(best).

Subject Nr.	Question number										Total score
	1	2	3	4	5	6	7	8	9	10	
subj01	4	4	4	4	4	3	3	3	4	4	92.5
subj02	4	4	4	4	4	4	4	4	4	4	100.0
subj03	3	2	3	1	3	4	4	3	3	4	75.0
subj04	4	3	4	4	3	2	4	3	4	4	87.5
subj05	4	4	3	4	3	4	4	4	3	4	92.5
subj06	2	4	4	2	4	3	4	4	2	4	82.5

3.6.4. Results of depth evaluation

The depth or in other words the distance between sensor and subject is a crucial factor for evaluating a physiotherapeutically correct knee bend. As mentioned in the introduction section (see chapter 1.3), the knee should not protrude beyond the toe tips during exercise performance. It is one of the four parameter which are tracked the whole time and provided as feedback for the subject.

An additional offline approach described in chapter 3.5.2 was used in order to observe the behavior of the programs depth minima. Figure 3.3 illustrates a topographical map of the recorded depth data limited by the KneeRoi. This particular figure shows an interpolated point cloud of both tracked legs of subject05 during a standing position.

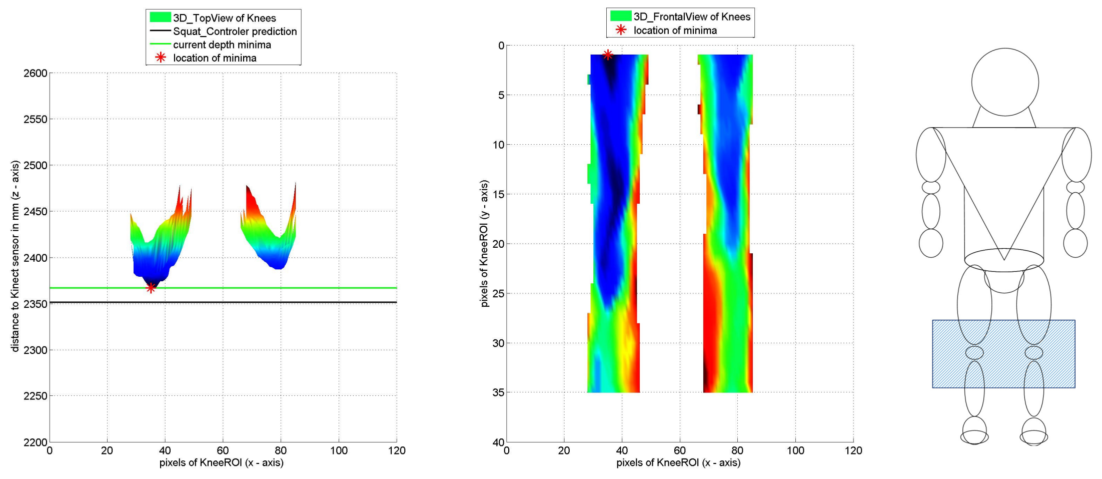


Figure 3.3: Topographical map, top (a) and frontal (b) view, of subject05 while in standing position.

The black line indicates the depth level of the particular frame provided as feedback during software execution (online approach). For direct comparison the green line marks the depth level obtained by the offline approach for the same frame. The red point in both subfigures indicates the exact location of the found minimum calculated by the offline analysis.

Figure 3.4 shows a similar topographical map with the only difference that the subject is in a kneeling position. A change in position can easily be recognized by comparing the depth levels of both figures (3.3 & 3.4).

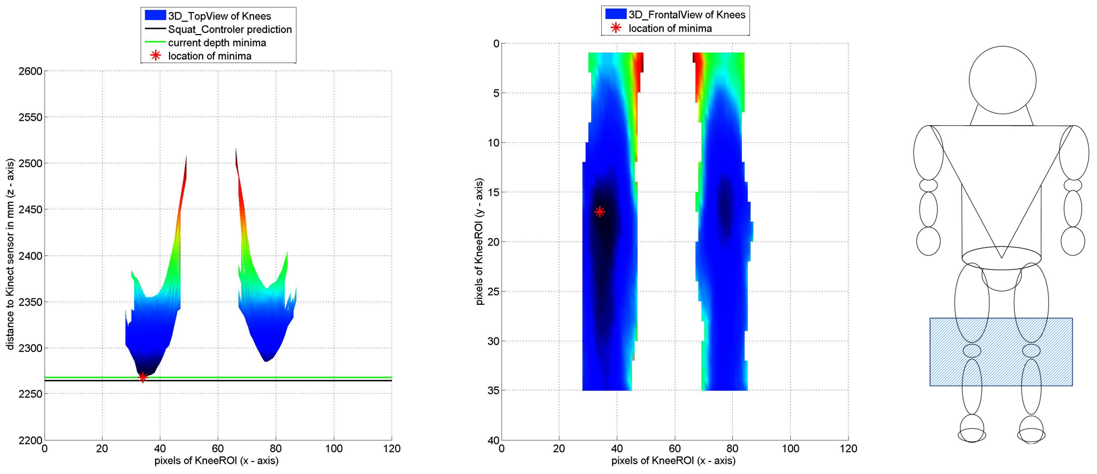


Figure 3.4: Topographical map, top (left) and frontal (right) view, of subject05 while in kneeling position.

The deviation between the two depth minima of the same frame but with different calculation methods is named "error" in this thesis and can be calculated like described in formula 3.2. All observed error values for each frame and each measurement were combined in order to calculate the mean value and the corresponding 10th percentile and 90th percentile for each subject. Table 3.4 summarizes the yielded results. Additionally it lists the $dist_{80\%}$ parameter, which describes the behavior of 80% of the error values.

By averaging over all subjects a mean error value of 14 mm was obtained and 80% of data lie within a mean distance of 65 mm.

Table 3.4: Mean values over 3 measurements for each subject.

Subject Nr.	$mean(error)$ mm	$prct_{10}(error)$ mm	$prct_{90}(error)$ mm	$dist_{80\%}$ mm
subj01	15	-29	59	88
subj02	20	-16	59	75
subj03	25	-14	66	81
subj04	5	-24	28	52
subj05	9	-14	37	51
subj06	12	-7	37	43
over all	14	-17	48	65

4. Discussion, Conclusion and a future outlook

The following sections discuss the obtained results, indicate the softwares limits, draw a conclusion and finally provide a future outlook.

4.1. Observations of the test measurements

In order to find a suitable value for the AREA_THRESHOLD parameter a total number of three measurements for each subject were performed. This particular parameter alters the boundaries of the feedback regarding the area parameters. Therefore the lower the value is, the more challenging it will be to remain within the boundaries (green area) while performing a squat. The results in table 3.2 supports this notion. While with an AREA_THRESHOLD of 150 just 47.5 % of all performed squats regarding the Area2 parameter were within the green area of the feedback visualization, the percentage rises to approximately 80% for AREA_THRESHOLD = 175 and 200.
Even though only six subjects were recorded and therefore no proper statistical statement is possible, the results indicates that a usage of an AREA_THRESHOLD of 175 would be best out of the three. On the one hand it still provides challenging boundaries but on the other hand does not lead to frustration of the participant.

Two different methods for calculating the minimum depth value for each frame were used in the course of this thesis. One of them was applied during the measurement by averaging over the two smallest values of the current frame (see chapter 2.2.2). The other method was performed after the actual measurement by using the recorded depth data (see chapter 3.5.2). The offline approach is considered as a more accurate estimation of the real depth due to the usage of a median filter and thus used as a reference.

The reason why the SquatController software uses a different evaluation approach is the increase in calculation complexity by using a median filter. With such a median filter each pixel value will be swapped with the median of the 8 neighboring pixels. Thus outliers are efficiently suppressed. But with those advantages comes a hefty increase of calculation complexity, which would have effected the thread stability of the SquatController. By using the trimmed approach described in chapter 2.2.2 a total frame rate of 4 frames per second can be achieved.

The results listed in table 3.4 describe the deviation between the two depth minima of the same frame, obtained by these two different calculation methods. Overall a mean error value of 14 mm was obtained and 80% of the data lie within a region of 65 mm. Even if a distance of 65mm seems to be a trifle too large, the results met the initial expectations based on a preliminary measurement which is summarized in the appendix chapter A.2.

One of the main influences for an increase of the calculated error would be the used moving average filter mentioned in chapter 2.2.3. The filter was used in order to smooth the response of the visualized feedback. Thus the subject can concentrate on its assignment and is not negatively influenced by a fast fluctuating visualization. The downside of applying this filter is the modification of the actual feedback value, which automatically leads to a larger fluctuation margin and therefore to a higher $dist_{80\%}$ value.

4.2. Usability of the SquatControler software

A total number of 6 subjects were asked to undergo a usability test. The results evaluation (table 3.3) yielded an averaged total score value of 88,33 of a total achievable score of 100 points.

A closer look on the individual scores of each question indicated that an initial instruction of the software by a technical person would be appreciated. However, the participants stated that just little instructions are required for the appropriate usage and furthermore supported the notion that most people would quickly learn to use the system.

4.3. Software and hardware limits

While the low costs and the advantage of a marker less sensing technology support the usage of the Kinect sensor for tele-rehab based software solutions, there are some limitations.

As it was the same with the first version of the Kinect sensor, it is necessary to clear the surroundings from objects, which could deceive the tracking algorithm leading to an incorrect measurement.

While depth levels up to 8 meters are detectable, the proper functionality of the SDK's skeleton tracking algorithm is guaranteed within 4.5 meters . With a horizontal angle of 70° and a vertical angle of about 43° a FOV is given (see figure A.2). Thus all tracking applications are limited by this region.

The Kinect is originally used as a game controller. Therefore the skeleton tracking algorithm provided by the SDK of Microsoft can only properly track each of a subject's body joints when the subject is directly facing the camera. Due to the fact that most games with the Kinect sensor respond to motion pattern rather than exact joint locations, it is sufficient for games, but prove to be a challenge if used for the tracking of rehab based applications.

In case of the developed SquatController software a specific routine checks whether the subject is facing the Kinect or not. Therefore overlapping skeleton joints can be avoided. However, the SquatController supports just one body to be calibrated and evaluated by the software, while the Kinect would be capable of tracking up to 6 bodies.

Another limit of the SquatController is that the software automatically observes the first appearing body and tries to perform the mentioned calibration. In the inconvenient case that the first appearing body would be the physiotherapist who is helping the actual participant, the SquatController will malfunction.

4.4. Conclusion

Aim of this study was the creation of a helpful software solution for a rehab training tool in order to track a knee bend or squat exercise and provide reliable feedback.

In order to answer the question of research the following statements can be made:

With an averaged total score of 88,33 out of 100 points (over all subjects), the usability evaluation of the programed software surpassed initial expectations. While initial instructions are recommended, the SquatControler software was assessed to be intuitive and easy to handle.

Furthermore, a comparison between the depth values determined by the SquatController and the depth values determined by a modified offline approach was performed. The comparison yielded an average deviation of 65mm over all subjects which represents an expected outcome under the consideration of measuring a moving subject with the Kinect sensor.

Both performed tests as well as the observations regarding the AREA_THRESHOLD parameter, yielded results which support the ability to track the knees during exercise performance and furthermore the usage of the SquatController as a rehab training tool. However, in order to bring the software to the next level, source modifications and further tests with more participants are advised.

4.5. Future outlook

While the SquatController proved to be a useful tool, there is still space for further improvements. It is advised that in the future, related researches should focus on a proper management and preparation of the recorded data. Thus if the executing Pc is connected with the Internet, an easy data sharing between user and physiotherapist would be possible. This feature is necessary to fulfill a basic requirement of an actual home-based training tool, which is not yet implemented in the current version of the SquatController.

Additional refinements regarding the depth determination algorithm are thinkable in order to increase the accuracy. A possible alternative would be a gradient decent approach with the initial condition of the original location of the knee joints provided by the SDK. The usage of such an algorithm would possibly decrease the amount of used loop cycles and therefore decrease the processor load.

Another crucial point of the to-do-list of the future would be to carry out more evaluations with a greater subject pool in order to gain proper statistical statements and therefore a better insight in the ability of the SquatController software.

4.5.1. Vision of Kinect4Rehab

There is much to do concerning the Kinect4Rehab project, mentioned in chapter 1.4. Before some modifications on the SquatController can be made in order to serve as a squat module for the project, initial thoughts of the main programs structure have to be made. Additionally a standard regarding the interfaces of the exercise modules must be developed. Only with such a standardized interface a similarly integration of each of the various modules is possible.

For the actual creation of the squat module, it should be possible to use the existing source code of the background working thread of the SquatController. Only the mentioned modifications of the standardized interface must be implemented.

Even if the formulated aim of this thesis does not hint a further usage of the program as a possible module for a greater project, it was indeed considered while the programming was done.

The great vision is the realization of a standardized software, capable of tracking a subjects physiotherapeutic exercise and provide reliable feedback. Additionally it is intended that the user has the opportunity to choose an exercise by loading an external exercise module. Furthermore a connection between the end user and a specialist is intended. With such a connection a physiotherapist would be able to track the users progress, modify the users training program by remote control and allow the user to train at home.

One of the basic concepts of the project is to provide the software under the GNU public license in order to allow a free usage and open the doors for other 3rd party programmers. This concept would benefit the development of all the thinkable modules regarding the various physiotherapeutic exercies.

References

[1] Abdur Rahman M, Qamar A M, Ahmed M a, Ataur Rahman M, Basalamah S. Multimedia interactive therapy environment for children having physical disabilities. *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval - ICMR '13* page 313 (2013)

[2] Angaran D M. Telemedicine and telepharmacy: current status and future implications. *American Journal of Health-System Pharmacy* 56(14):1405–1426 (1999)

[3] Blyth W J, Blyth M M. *Telecommunications: Concepts, Development, and Management.* Glencoe/McGraw-Hill School Publishing Company (1990). P. 280 - 282

[4] Brooke J. SUS - A quick and dirty usability scale. *Usability evaluation in industry* 189(194):4–7 (1996)

[5] Capecci M, Ceravolo M G, D'Orazio F, Ferracuti F, Iarlori S, Lazzaro G, Longhi S, Romeo L, Verdini F. A tool for home-based rehabilitation allowing for clinical evaluation in a visual markerless scenario. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-November:8034–8037 (2015)

[6] Chang Y J, Chen S F, Huang J D. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities* 32(6):2566–2570 (2011)

[7] Chapinal Cervantes J, Vela F L G V, Rodríguez P P. Natural interaction techniques using Kinect. *Proceedings of the 13th International Conference on Interacci{ó}n Persona-Ordenador - INTERACCION '12* pages 1–2 (2012)

[8] Cheng P T, Wu S H, Liaw M Y, Wong A M K, Tang F T. Symmetrical body-weight distribution training in stroke patients and its effect on fall prevention. *Archives of Physical Medicine and Rehabilitation* 82(12):1650–1654 (2001)

[9] Gossman J. Introduction to Model/View/ViewModel pattern for building WPF apps. *Viitattu: 1410 2013 Saatavissa: `http://blogsmsdncom/b/johngossman/archive/2005/10/08/478683aspx`* (2005)

[10] Gotsis M, Tasse A, Swider M, Lympouridis V, Poulos I C, Thin A G, Turpin D, Tucker D, Jordan-Marsh M. Mixed reality game prototypes for upper body exercise and rehabilitation. *2012 IEEE Virtual Reality (VR)* pages 181–182 (2012)

[11] Griebler R, Anzenberger J, Eisenmann A. *Herz-Kreislauf-Erkrankungen in Österreich* (2014)

[12] IFixit Cooperation. Xbox One Kinect Teardown. `https://de.ifixit.com/Teardown/Xbox+One+Kinect+Teardown/19725`

[13] Khoshelham K, Elberink S O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* 12(2):1437–1454 (2012)

[14] Lange B, Chang C Y, Suma E, Newman B, Rizzo a S, Bolas M. Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor. *Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology SocietyIEEE Engineering in Medicine and Biology SocietyConference* 2011:1831–1834 (2011)

[15] Lange B, Koenig S, McConnell E, Chang C Y Y, Juang R, Suma E, Bolas M, Rizzo A, McConnell E, Suma E, Bolas M, Rizzo A, Juang R, Chang C Y Y, Juang R, Suma E, Bolas M, Rizzo A. Interactive game-based rehabilitation using the Microsoft Kinect. *Ieee Virtual Reality Conference 2012 Proceedings* 34:170–171 (2012)

[16] Langhorne P, Coupar F, Pollock A. Motor recovery after stroke: a systematic review. *The Lancet Neurology* 8(8):741–754 (2009)

[17] Lun R, Zhao W. A Survey of Applications and Human Motion Recognition with Microsoft Kinect. *International Journal of Pattern Recognition and Artificial Intelligence* (January):49 (2015)

[18] Medina A. Three dimensional camera and range finder (1992)

[19] Medina A, Gayá F, del Pozo F. Compact laser radar and three-dimensional camera. *J Opt Soc Am A* 23(4):800–805 (2006)

[20] Mentiplay B F, Perraton L G, Bower K J, Pua Y H, McGaw R, Heywood S, Clark R A. Gait assessment using the Microsoft Xbox One Kinect: Concurrent validity and inter-day reliability of spatiotemporal and kinematic variables. *Journal of Biomechanics* 48(10):2166–2170 (2015)

[21] Microsoft Cooperation. Kinect for Windows SDK 1.0. `https://www.microsoft.com/en-us/download/details.aspx?id=28782`

[22] Microsoft Cooperation. Kinect for Windows SDK 2.0. `https://www.microsoft.com/en-us/download/details.aspx?id=44561`

[23] Microsoft Cooperation. The MVVM Pattern. `https://msdn.microsoft.com/en-us/library/hh848246.aspx`

[24] Mottura S, Fontana L, Arlati S, Zangiacomi A, Redaelli C, Sacco M. A virtual reality system for strengthening awareness and participation in rehabilitation for post-stroke patients. *Journal on Multimodal User Interfaces* 9(4):341–351 (2015)

[25] Nyberg L, Gustafson Y. Patient falls in stroke rehabilitation a challenge to rehabilitation strategies. *Stroke* 26(5):838–842 (1995)

[26] Ozturk A, Tartar A, Ersoz Huseyinsinoglu B, Ertas A H. A clinically feasible kinematic assessment method of upper extremity motor function impairment after stroke. *Measurement: Journal of the International Measurement Confederation* 80:207–216 (2016)

[27] Saini S, Rohaya D, Rambli A, Sulaiman S, Zakaria M N, Rohkmah S, Shukri M, Iskandar B S. A Low-cost Game Framework for a Home-based Stroke Rehabilitation System. In *International Conference on Computer & Information Science*, pages 55–60 (2012)

[28] Springer S, Seligmann G Y. Validity of the kinect for gait assessment: A focused review. *Sensors (Switzerland)* 16(2):1–13 (2016)

[29] Suma E A, Lange B, Rizzo A, Krum D M, Bolas M. FAAST: The flexible action and articulated skeleton toolkit. *Proceedings - IEEE Virtual Reality* pages 247–248 (2011)

[30] Tucker J K, Slp D. Perspectives of Speech-Language Pathologists on the Use of Telepr actice in Schools : Quantitative Survey Results. *International Journal of Telerehabilitation* 4(2):61–72 (2012)

[31] Valcik J, Sedmidubsky J, Zelzula P. Improving Kinect-Skeleton Estimation. *Computers & Graphics* 26(July):575–586 (2002)

[32] Yang L, Zhang L, Dong H, Alelaiwi A, El Saddik A. Evaluating and Improving the Depth Accuracy of Kinect for Windows v2. *IEEE Sensors Journal* PP(99):1 (2015)

[33] Zhao W, Feng H, Lun R, Espy D D, Reinthal M A. A Kinect-based rehabilitation exercise monitoring and guidance system. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS* pages 762–765 (2014)

[34] Zhao W, Lun R, Espy D D, Reinthal M A. Rule Based Realtime Motion Assessment for Rehabilitation Exercises. (2014)

A. Appendix

A.1. Used tools and software

`Kinect for Windows SDK v2 (v.2.0.1410.19000):`

The Kinect for Windows Software Development Kit (SDK) 2.0 enables developers to create applications that support gesture and voice recognition, using Kinect sensor technology on computers running Windows 8, Windows 8.1, and Windows Embedded Standard 8. [3]

`Visual Studio Enterprise 2015, (version 14.024720.00 Update 1):`

Visual Studio Express for Windows Desktop lets you take full advantage of Windows with XAML designers, a productive IDE, and a variety of programming languages including C#, Visual Basic, and C++. Choose between Windows Presentation Foundation (WPF), Windows Forms, and Win32, to target the Windows desktop with the right technology for your application and your skills. [4]

`Matlab R2013a (8.1.0.604), 64bit:`

It is a commercial software of Mathworks Inc., Natick, USA and provides a powerful tool to do numerical calculations with matrix manipulations. The software package is cross-platform programmed which means that it is available for different architectures and operating systems (Windows, Linux, Mac) and the license is proprietary.

`Import_SQC_Data.m`

Self scripted Matlab file used for importing raw data into the workspace of Matlab. It requires two .DAT files as input.

`Data_Evaluation.m:`

Self scripted Matlab file used for calculating a reference depth value from the original depth values provided by the depth stream of the Kinect sensor.

`Error_Evaluation.m:`

Self scripted Matlab file used for calculating the deviation between offline and online depth values.

`Useability_Evaluation.m :`

[3]obtained from: `https://www.microsoft.com/en-us/download/details.aspx?id=44561`
[4]obtained from: `https://go.microsoft.com/fwlink/?LinkId=691979&clcid=0x409`

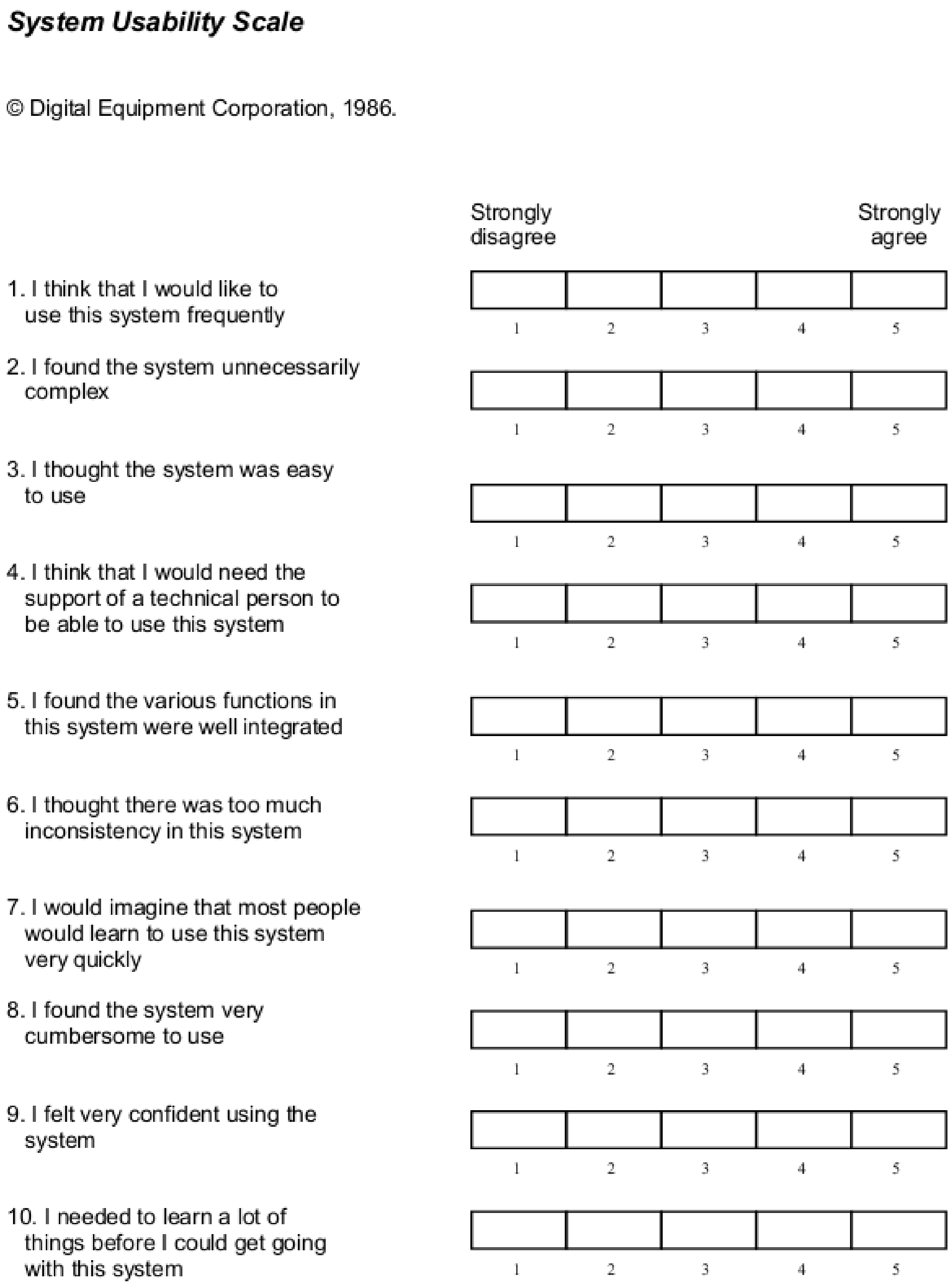
Self scripted Matlab file used for evaluating the usability of the SquatController software.

A.2. Kinect specific hardware requirements

For a proper usage of the Kinect sensor it is necessary to meet the following requirements (acquired from [22]):

- 64-bit (x64) processor
- 4 GB Memory (or more)
- Physical dual-core 3.1 GHz (2 logical cores per physical) or faster processor
- USB 3.0 controller dedicated to the Kinect for Windows v2 sensor
- DX11 capable graphics adapter

A.3. Usability test



System Usability Scale

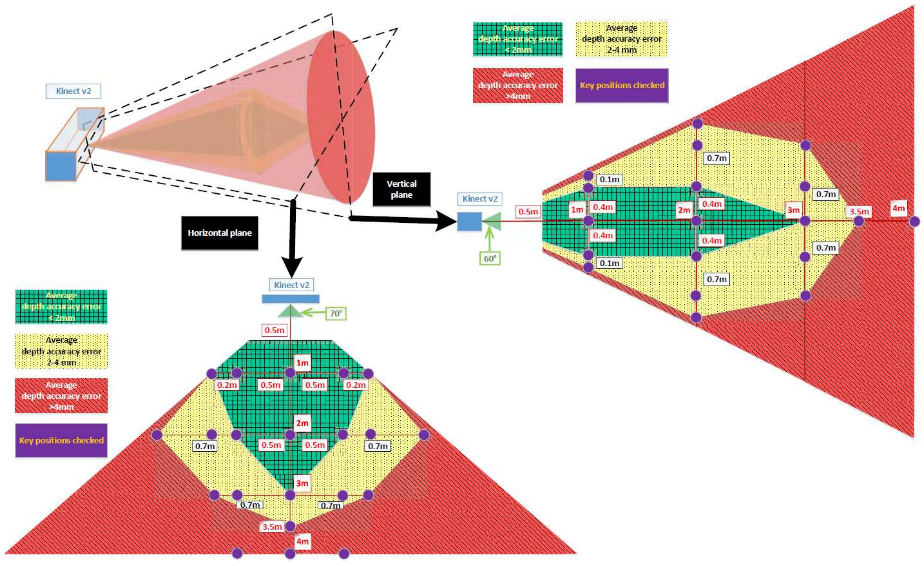
© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure A.1: Copy of used usability test obtained from [4]

A.4. Preliminary experiment of the precision of the Kinect v2

Based on an investigation done by Yang et al. [32], a preliminary experiment was performed. The research provides insight about the depth accuracy of the second version of the Kinect sensor. They described 3 different regions depending on the distance in respect to the sensor (for detailed specifications see figure A.2 at the appendix section). The first region exhibits a depth accuracy error of $< 2mm$, the second a depth accuracy error of $2 - 4mm$ and the third region exceeds the $4mm$ boundary. Those results are promising, though the investigation used static objects as reference.



Kinect v2
Vertical plane
Horizontal plane
Average depth accuracy error < 2mm
Average depth accuracy error 2-4 mm
Average depth accuracy error >4mm
Key positions checked
60°
70°
0.5m
1m
2m
3m
3.5m
4m
0.1m
0.2m
0.4m
0.5m
0.7m

Figure A.2: Accuracy error distribution of Kinect for Windows v2. Figure from [32]

Before the consideration of creating a rehab assisting tool with the Kinect, it was necessary to investigate, if the mentioned accuracy or in further respects the precision of the skeletal tracking algorithm, provided by Microsoft's SDK, is within a certain margin. Thus a preliminary experiment with 2 tasks targeting on the lower limb region was performed.

As a descriptive parameter the distances between minimum and maximum (d100) and the distances between prct10 and prct90 (d80) of the joint fluctuations were used. Reason for this particular description of the measured data were the different probability density functions (see figure A.3) of the fluctuating joints.

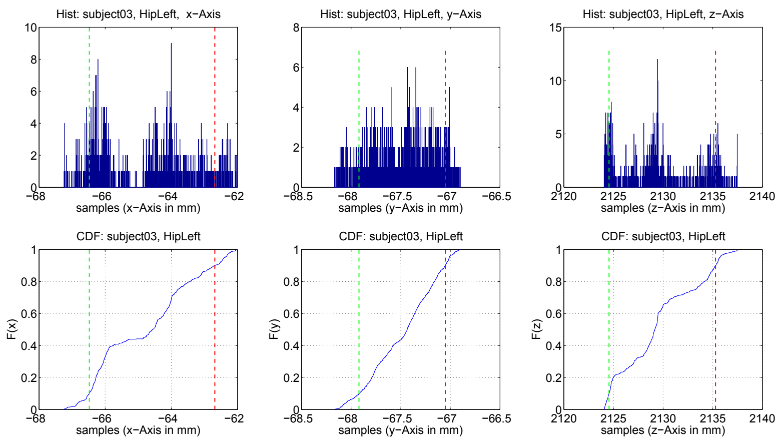


Figure A.3: Histogramm (upper row) and cdf (bottom row), x-/y-/z-axis (left to right), `HipLeft` joint, subject03

The obtained results of the first task, with the purpose on observing joint fluctuation of a still remaining subject over time, were promising. 95% of the results regarding the d80 parameter and 80% of the results regarding the d100 parameter were within the stated 10 mm margin.

Task 2 provided insight on the scale in which those distances differ, if there was a change in position. About 28% of the results regarding the d80 parameter and just 8% of the results regarding the d100 parameter were within the 10 mm margin.

Over all the results of the experiment leaded to the conclusion that Kinect v2 and its tracking algorithm should be sufficient as device for the development of rehab based exercises.

The whole result collection of the preliminary experiment with detailed information of all coordinates (x,y,z) are as follows:

	S01		S02		S03		S04		S05		S06	
	d80	d100	d80	d100	d80	d100	d80	d100	d80	d100	d80	d100
	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm
LeftHip	2	3	2	3	4	5	2	3	1	1	2	5
LeftKnee	2	3	1	2	3	5	2	3	1	2	5	9
LeftAnkle	1	2	1	2	2	3	1	2	1	2	1	2
LeftFoot	2	32	1	2	1	3	2	4	2	3	2	21
RightHip	2	3	2	3	3	5	3	4	1	2	2	5
RightKnee	1	2	1	2	1	2	3	5	1	2	2	4
RightAnkle	1	2	1	1	1	2	1	2	1	2	1	2
RightFoot	4	13	1	2	2	5	2	5	1	2	2	32

	S07		S08		S09		S10		S11	
	d80	d100	d80	d100	d80	d100	d80	d100	d80	d100
	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm
LeftHip	2	3	1	2	5	9	2	4	2	5
LeftKnee	6	11	2	3	3	5	1	2	1	2
LeftAnkle	1	1	1	1	1	3	1	1	1	2
LeftFoot	1	3	2	3	2	4	1	3	1	2
RightHip	2	3	1	2	5	8	2	3	3	5
RightKnee	1	3	1	1	2	4	1	4	2	4
RightAnkle	1	2	1	2	2	10	1	1	1	2
RightFoot	1	3	2	4	3	9	1	2	1	2

Table A.1: d80 and d100 of task 1, x-axis, subject01 to subject11

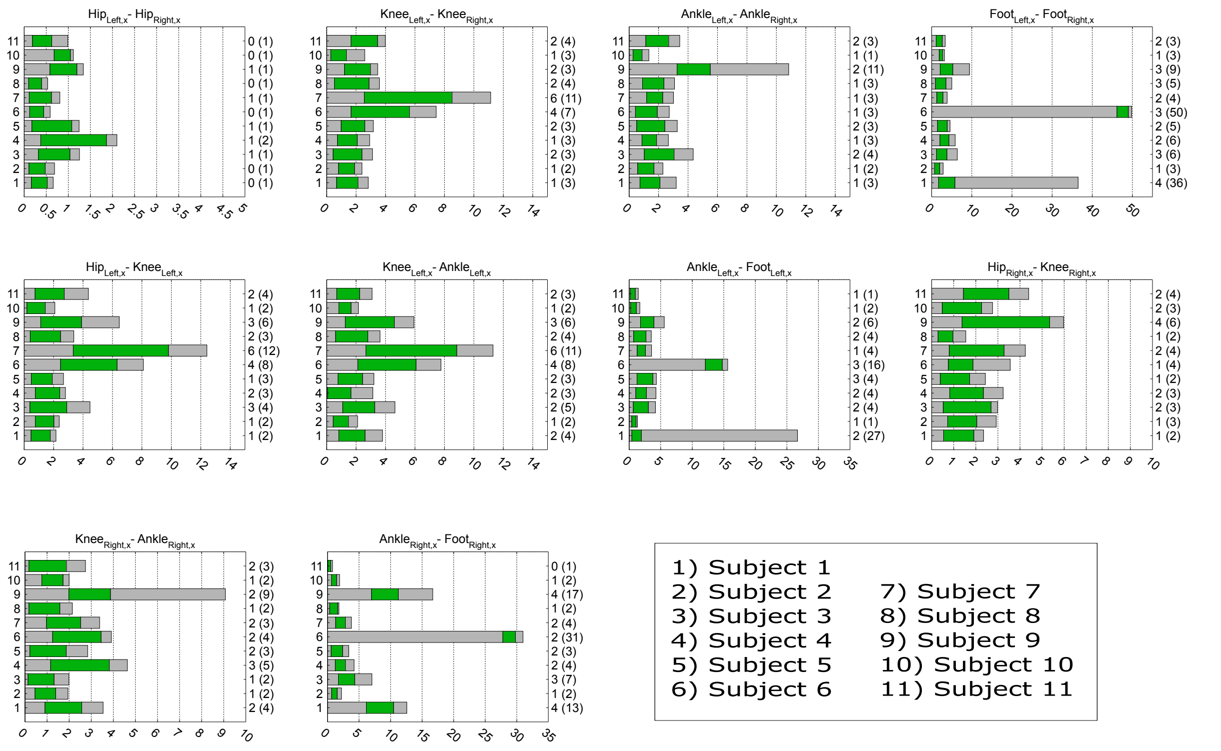


Figure A.4: d80 (green) and d100 (grey) in millimeters, x-axis, task 1, in various joint combinations

	S01		S02		S03		S04		S05		S06	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	1	3	1	2	1	1	3	5	1	2	1	2
LeftKnee	2	3	2	5	2	5	3	8	2	6	2	5
LeftAnkle	2	3	3	4	2	7	3	5	4	6	2	5
LeftFoot	2	47	2	4	1	5	3	6	2	5	1	52
RightHip	1	3	1	1	2	2	3	5	2	3	1	2
RightKnee	8	14	2	4	2	4	6	12	6	11	11	20
RightAnkle	3	8	2	4	2	4	2	4	3	6	2	5
RightFoot	3	52	2	3	2	4	3	7	2	4	2	52

	S07		S08		S09		S10		S11	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	1	2	2	2	3	4	1	5	1	2
LeftKnee	4	8	7	12	6	16	1	3	3	8
LeftAnkle	2	5	2	3	4	9	3	7	4	8
LeftFoot	2	5	1	2	2	5	1	48	4	8
RightHip	1	1	2	3	4	5	1	4	2	3
RightKnee	11	21	2	6	6	16	3	6	6	16
RightAnkle	3	7	4	8	4	11	1	3	2	3
RightFoot	2	4	3	7	3	7	1	3	1	3

Table A.2: d80 and d100 of task 1, y-axis, subject01 to subject11

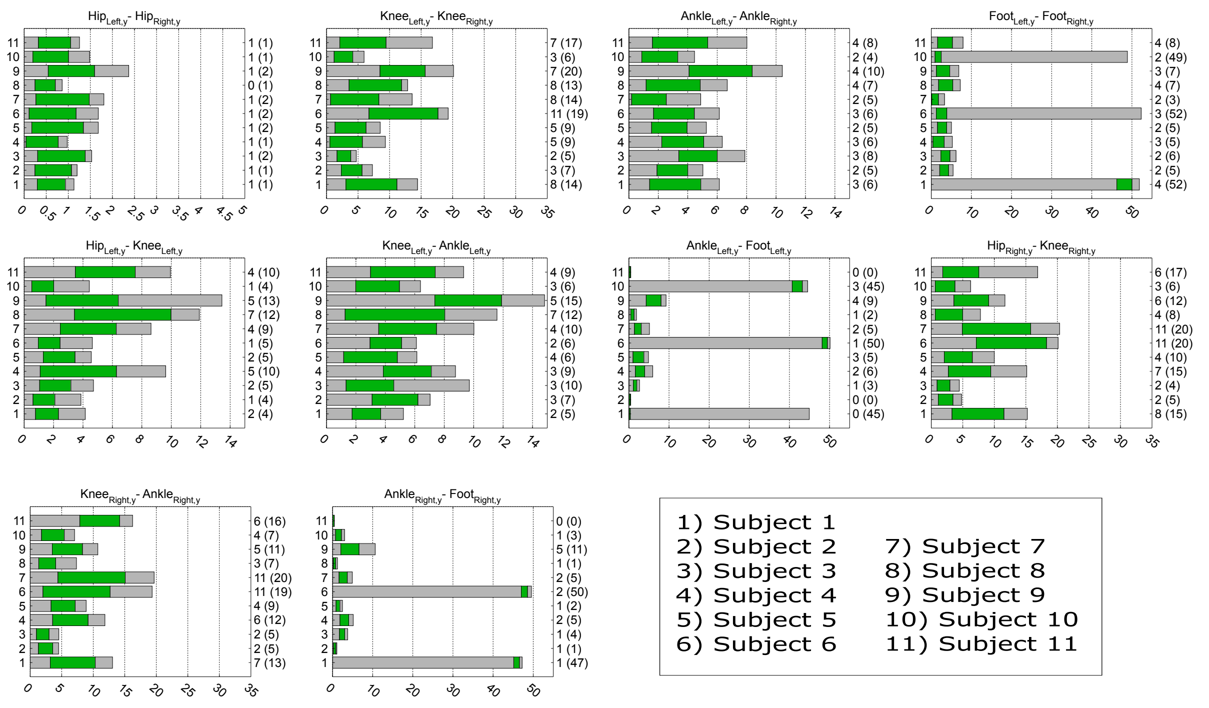


Figure A.5: d80 (green) and d100 (grey) in millimeters, y-axis, task 1, in various joint combinations

	S01		S02		S03		S04		S05		S06		S07		S08		S09		S10		S11	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	12	18	7	11	11	13	9	14	10	12	4	10	26	31	10	16	12	15	10	16	15	23
LeftKnee	7	11	4	6	6	8	6	10	5	7	1	4	12	17	3	8	5	7	4	6	10	15
LeftAnkle	2	4	1	2	2	3	2	4	2	4	1	2	2	4	1	2	1	4	1	3	3	4
LeftFoot	2	86	1	2	2	5	4	10	3	6	2	81	2	7	1	4	4	10	1	59	3	4
RightHip	12	18	7	11	9	13	9	14	11	13	4	10	24	31	7	13	10	13	11	17	17	23
RightKnee	8	13	4	5	7	9	6	10	6	7	3	8	14	19	3	6	5	8	2	4	8	12
RightAnkle	3	5	1	2	2	3	1	3	1	2	1	2	3	6	2	4	2	6	1	2	1	3
RightFoot	4	78	1	3	2	5	4	13	2	4	4	92	2	5	2	4	4	12	1	3	1	3

Table A.3: d80 and d100 of task 1, z-axis, subject01 to subject11

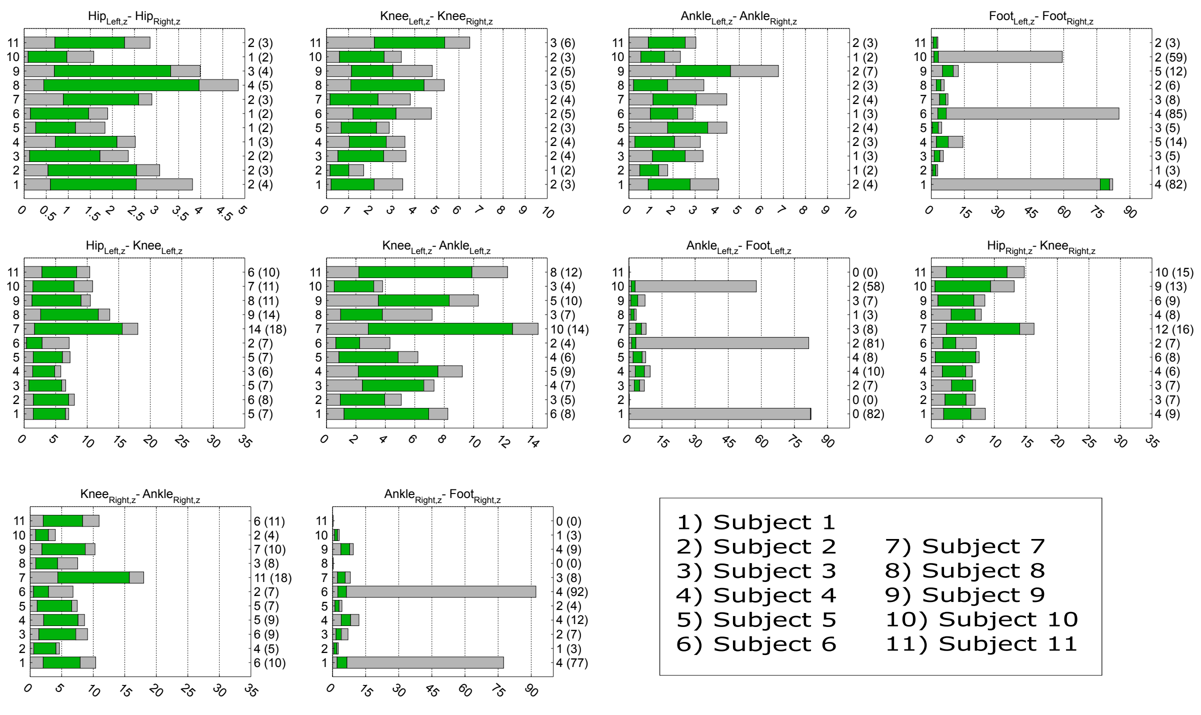


Figure A.6: d80 (green) and d100 (grey) in millimeters, z-axis, task 1, in various joint combinations

	S01		S02		S03		S04		S05		S06	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	12	22	23	41	12	36	15	23	13	20	10	19
LeftKnee	14	27	26	56	17	29	13	22	18	26	14	19
LeftAnkle	9	56	24	39	5	7	9	22	7	11	2	3
LeftFoot	18	57	21	43	10	37	10	41	4	9	4	8
RightHip	12	24	19	45	12	30	16	26	13	26	15	27
RightKnee	12	17	40	64	11	19	22	28	17	35	6	12
RightAnkle	9	28	26	43	4	8	8	20	6	8	18	24
RightFoot	11	73	23	33	8	58	6	29	4	7	18	31

	S07		S08		S09		S10		S11	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	18	35	22	44	15	34	23	44	12	28
LeftKnee	7	20	18	35	20	37	10	17	10	18
LeftAnkle	4	9	9	23	15	24	3	5	3	6
LeftFoot	7	37	9	48	18	46	3	7	4	7
RightHip	15	36	17	43	20	42	25	46	14	31
RightKnee	14	29	16	25	31	53	7	16	14	23
RightAnkle	7	19	6	13	14	19	4	6	5	7
RightFoot	8	34	29	47	22	64	8	20	33	41

Table A.4: d80 and d100 of task 2, x-axis, subject01 to subject11

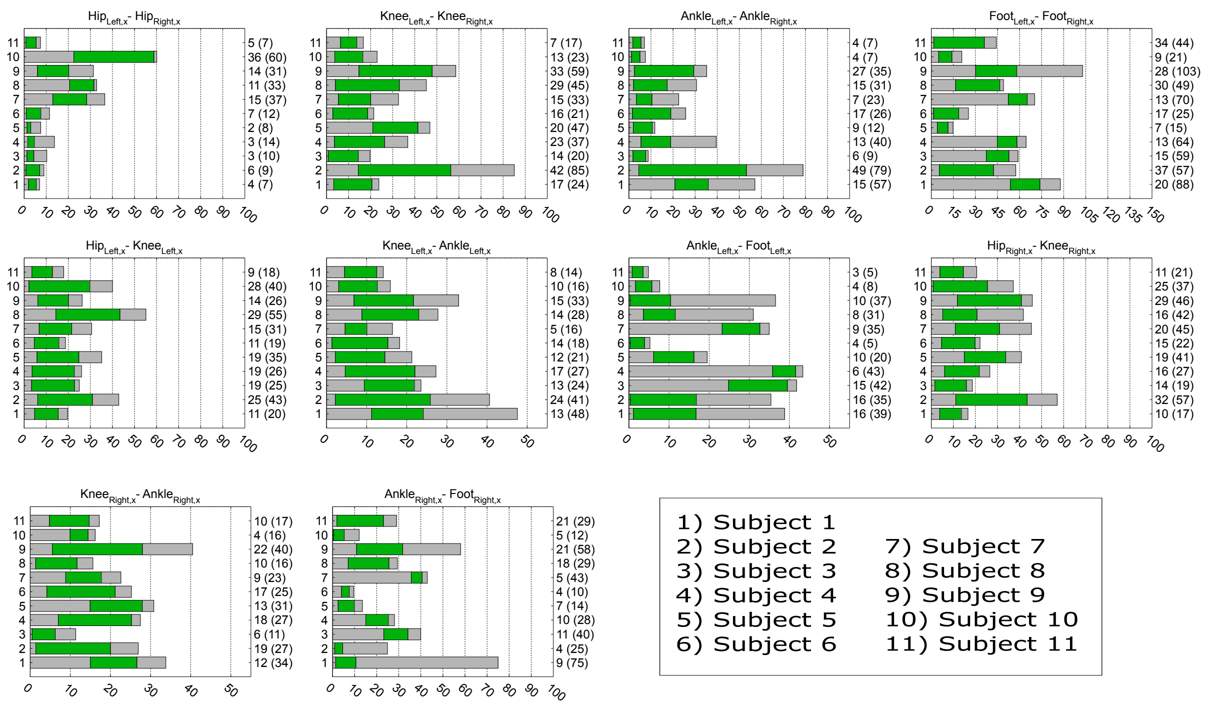


Figure A.7: d80 (green) and d100 (grey) in millimeters, x-axis, task 2, in various joint combinations

	S01		S02		S03		S04		S05		S06	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	21	59	33	79	17	31	62	120	22	36	21	34
LeftKnee	9	19	53	76	8	22	18	44	9	20	22	37
LeftAnkle	39	145	32	60	5	9	15	22	3	7	3	8
LeftFoot	69	163	32	56	5	27	22	42	5	11	5	10
RightHip	17	53	29	75	18	33	61	122	21	34	21	33
RightKnee	10	21	54	71	9	25	21	48	10	19	16	26
RightAnkle	39	153	28	54	4	7	17	32	3	7	68	77
RightFoot	78	181	17	33	6	23	30	63	6	12	67	77

	S07		S08		S09		S10		S11	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	68	222	91	270	63	118	31	151	38	95
LeftKnee	26	111	23	137	14	36	13	107	21	49
LeftAnkle	11	59	12	52	37	46	6	14	3	10
LeftFoot	12	59	20	61	37	71	7	28	3	13
RightHip	68	224	95	279	59	123	32	149	36	92
RightKnee	20	116	26	132	15	38	11	87	16	28
RightAnkle	7	54	8	47	34	54	6	18	7	13
RightFoot	12	45	14	67	27	51	27	46	30	43

Table A.5: d80 and d100 of task 2, y-axis, subject01 to subject11

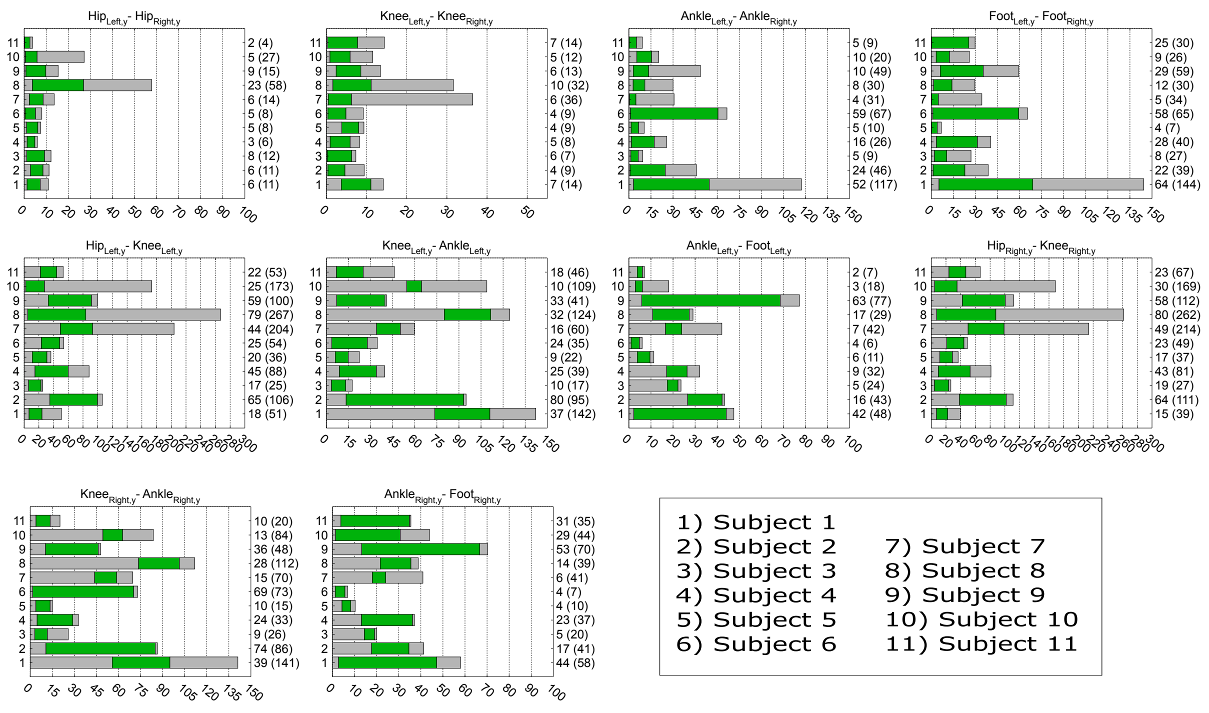


Figure A.8: d80 (green) and d100 (grey) in millimeters, y-axis, task 2, in various joint combinations

	S01		S02		S03		S04		S05		S06	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	15	35	44	63	19	44	44	65	26	51	35	81
LeftKnee	19	34	60	88	20	40	33	45	21	38	25	40
LeftAnkle	18	93	41	53	8	19	24	59	8	11	8	18
LeftFoot	108	236	103	123	9	116	6	123	13	26	9	18
RightHip	16	30	44	62	22	53	46	69	24	49	36	76
RightKnee	18	31	57	94	25	45	37	51	20	40	23	43
RightAnkle	25	122	44	63	7	14	36	67	7	11	68	79
RightFoot	111	256	90	112	15	108	16	118	10	29	68	79

	S07		S08		S09		S10		S11	
	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm	d80 mm	d100 mm
LeftHip	48	86	36	178	36	80	43	113	28	53
LeftKnee	43	88	47	130	39	93	30	82	27	55
LeftAnkle	11	40	13	56	59	79	8	22	7	14
LeftFoot	16	102	66	149	138	203	6	18	9	14
RightHip	51	85	44	202	39	85	43	113	27	49
RightKnee	44	84	41	124	38	99	34	83	29	50
RightAnkle	20	41	16	52	60	83	17	37	8	14
RightFoot	14	102	78	106	53	155	79	93	102	109

Table A.6: d80 and d100 of task 2, z-axis, subject01 to subject11

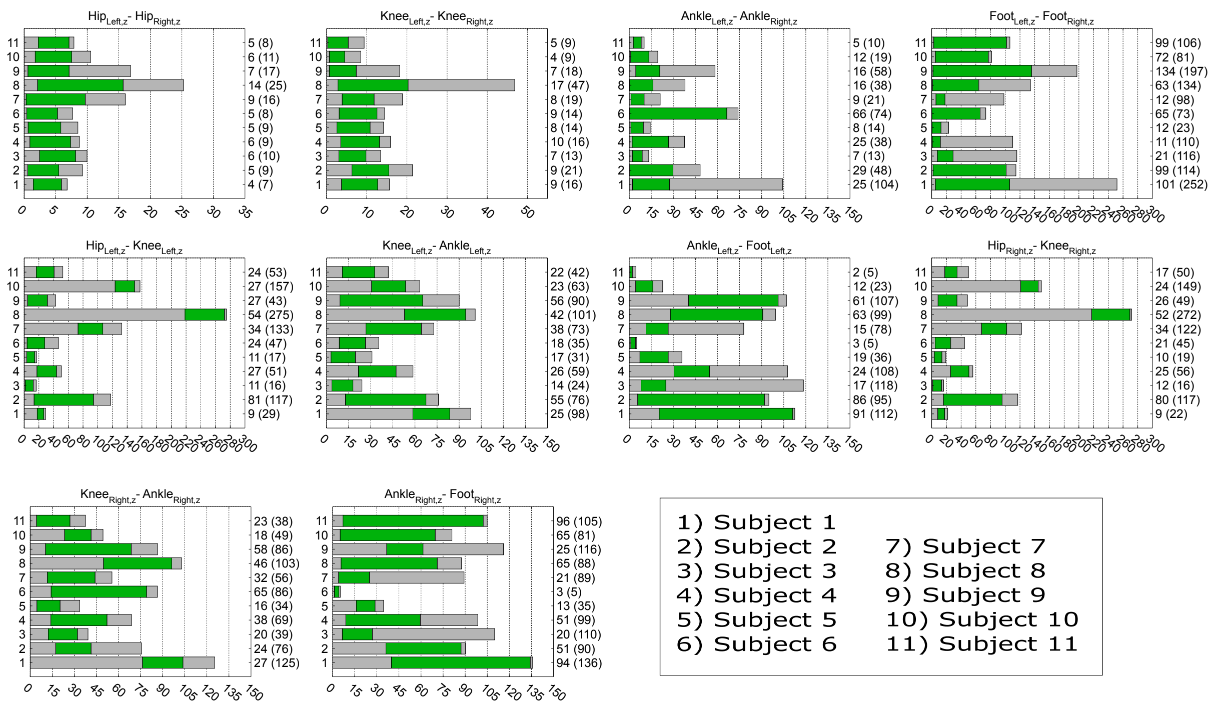


Figure A.9: d80 (green) and d100 (grey) in millimeters, z-axis, task 2, in various joint combinations