NTNU

Norwegian University of
Science and Technology

Automatically Quantifying Human Activity in Design Activities

With Microsoft Kinect Depth Sensors

Leif Erik Bjørkli

Master of Science in Engineering and ICT
Submission date: September 2015
Supervisor: Martin Steinert, IPM

Norwegian University of Science and Technology
Department of Engineering Design and Materials

Abstract

Observing design activities can give insights that can be used to improve the activities and give feedback to participants. In the past the process of observing and analyzing design activities was done manually with video which is very resource demanding. With the launch of the Microsoft Kinect 3D depth sensor and later an improved version in 2014, automatically tracking human behavior became much easier. The thesis evaluates the suitability for the Kinect sensor to contribute to automating design observations and give real-time feedback to participants. A thorough analysis of the Kinect v1 and v2 sensors is conducted. A proof of concept for an observation setup and a sensing application is developed. The sensing application tracks participants from a ceiling mounted position and estimates real-time mechanical energy-use and creates heat maps of their movements. A prototype of a design experiment is developed to test the application and observation setup. The results show that, except for some challenges inherent to the depth sensing technology, the Kinect v2 sensor shows great potential for contributing to automating design observation setups. Finally, issues to be aware of when using Kinects specifically in design observation and Design Observatories are highlighted. Future work involves developing a permanent design observatory with Kinect sensors and exploring how the audio capabilities and gesture analysis software for the Kinect can be used in design observation.

Norwegian Abstract

Observasjon av design aktiviteter kan gi innsikt som kan brukes til å både å forbedre selve aktiviteten og gi tilbakemelding til deltakere. Tidligere har observasjon og analyse av design aktiviteter blitt gjennomført manuelt ved hjelp av video, noe som er svært ressurskrevende. Da en oppgradert versjon av Microsofts Kinect 3D dybdesensor ble lansert i 2014, ble automatisk deteksjon og analyse av menneskers bevegelser mye enklere. Denne masteroppgaven vurderer hvor godt egnet Kinect sensoren er til å automatisere design observasjon og kunne gi sanntid tilbakemelding til deltakere. En grundig analyse av Kinect v1 og Kinect v2 ble gjennomført og et proof-of-concept for en observasjonsmetode og sensing-applikasjon ble utviklet. Sensing-applikasjonen detekterer og analyserer deltakere fra en takmontert posisjon og estimerer sanntid mekanisk energibruk og lager varmekart over bevegelsene til deltakerne. En prototype av et design eksperiment ble utviklet til å teste applikasjonen og observasjons-oppsettet. Resultatene viser at Kinect v2 sensoren kan potensielt bidra til å betydelig automatisere design observasjon, til tross for at det er noen feilkilder knyttet til dybdemålingsteknologien. Avslutningsvis er mulige utfordringer ved bruk av Kinect sensorer spesifikt i rom for design observasjon diskutert. Videre arbeid innebærer å bygge et permanent rom for observasjon av design aktivititer med Kinect sensorer, i tillegg til å utforske hvordan dens egenskapene for stemmegjennkjenning og analyse av gester kan bli brukt i design observasjon.

Preface

This master project was conducted as part of the master program Engineering and ICT with a specialization in Product Development and Material Science at the Norwegian University of Science and Technology. The project was written in the spring/summer 2015.

I would like to thank Matti Hämäläinen at the Sino-Finnish Center at Tongji University Shanghai for being a great facilitator and giving advice one the master thesis process. He provided me with the equipment I needed for my thesis and gave me full access and freedom at the Sino-Finnish Center from day one.

Finally, I would like to thank my supervisor Martin Steinert for inspiring guidance and great ideas for the thesis.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

SDK	Software Development Kit
AdaBoost	Adaptive Boosting algorithm
RFR	Random Forest Regression algorithm
IR	Infrared
ToF	Time-of-flight
API	Application Programming Interface

1 Design Observation

1.1 Background

The term early-stage conceptual design describes the phase in a project where idea generation and exploring ideas are being done, for instance brainstorming and early prototyping.

John Tang at the Center for Design Research at Stanford University pioneered in 1989 analysis of collaborative design activities. He wanted to better understand the collaborative workspace activities so the design process could be improved and also to support design of tools for improving workspace activities. A framework for analyzing workspace activity was proposed, which became a foundation for the Observe-Analyze-Intervene cycle later used in Design Observatories (J. C. Tang, 1989)(J. C. Tang & Leifer, 1988)(Törlind et al., 2009). In Tang & Leifer (1988) design activities in conceptual design phases were investigated. Sessions of teams of 3-4 were videotaped, from which transcripts with annotations were made.

In Tang & Leifer (1991) the collaborative drawing activity of design teams was studied using video-based interaction analysis methods. Interaction analysis is a qualitative analysis method traditionally used in social sciences that integrates an ethnographic perspective with fine grained analysis of human interaction. A crucial element of the interaction analysis approach is that the participants should be observed in their natural environment without any intrusion from observers. Eight different sessions of teams of 3-4 people working on conceptual design tasks were observed and recorded. One camera was aimed at the workspace and one wide-angle camera captured the whole group. Later, the recordings were transcribed and analyzed. The study provided several new and useful insights into design activities, as well as showing some limitations to the approach of video-based analysis. The tedious work of creating transcripts and annotating video recordings limited the approach to only be able to analyze shorter time spans. In addition, an issue with how the findings should be generalized to be useful outside the context of the studies was highlighted.

Minneman et al. (1995) continued the work of Tang (1989) and presented several tools, that he collectively called Coral, to capture and facilitate analysis of collaborative activities. The aim was to speed up the process of revising the recordings by methods of indexing recordings and retrieving the indexes. In 2002, the first permanent design bservatory was built at the Center for Design Research at Stanford University (Carrizosa, Eris, Milne, & Mabogunje, 2002). The design observatory built on the work of Tang (1989) and sought to make the process of analyzing design activities more efficient and consistent by storing data digitally, making

synchronous viewing of several camera views possible and eliminating the work of setting up and taking down equipment.

Törlind et al. (2009) in "Lessons Learned and Future Challenges for Design Observatory Research" discussed previous research on environments designed specifically for design team observation. The paper argues several implications for future design observatories: While design observatories in the past focused on observation, real-time analysis will be possible in future observatories and thereby the possibility of intervening to improve the design activity. The coding schemes should be robust and automated by machines so capturing data for longer periods of time is possible. Design observatories should support an iterative research approach, in order to allow the researchers to iterate over the setup, questions and coding scheme.

Recently, Dinar et al. (2015) reviewed empirical research of designer thinking from the last quarter century, and concluded that future studies may need to apply computer based data collection and automated analyses. As argued in Törlind et al. (2009) and Dinar et al. (2015), one of the main areas for potential improvement in empirical research of design activities is automating the collection and analysis of empirical data.

Traditionally, normal RGB cameras have been used for capturing and analyzing design activities. Extensive research has been done the past decades in the field of traditional computer vision. Still, some major challenges in analysis of human behavior in RGB data remain unsolved due to the nature of the data (Santhanam, Sumathi, & Gomathi, 2012). The recent years, interesting advances in sensor technology have become accessible to researchers with the Microsoft Kinect sensor (Z. Zhang, 2012). One of the goals of this thesis is to explore how the Kinect v1 sensor and the new Kinect v2 sensor can contribute to automating data capture and analysis in design activities.

1.2 Goals of Thesis

1. ***Develop a thorough understanding of the advantages and disadvantages of the new Kinect v2 sensor compared to the old Kinect v1.***

   Microsoft released a new and updated version of their Kinect sensor in July 2014. The old sensor has gained great popularity among researchers in several research fields. A goal with this thesis is to identify the advantages and possible disadvantages with the new sensor, with the perspective of sensing human behavior in design activities.

2. ***Explore how Kinect sensors can be used to non-intrusively automate data capture and analysis of design activities.***

   Suggest a proof of concept on how the Kinect sensor can contribute to automating data capture and analysis in empirical research on design activities. In addition, point out possible pitfalls and opportunities of using the sensor.

3. ***Explore how human activity can be quantified with the Kinect sensor.***

   Recent research from Stanford University suggest that walking boosts creative ideation in real-time and shortly after, especially in the expression of associative memory and creative divergent thinking (Oppezzo & Schwartz, 2014). Further, physical activity in general has been shown to boost specific cognitive processes (Brisswalter, Collardeau, & René, 2012)(Tomporowski, 2003). A goal of this thesis is to explore how activity and energy can be quantified in design activities, as inspiration to further research on how energy-use and human activity influence design activities.

2 Non-Intrusive Sensing

Topics such as sensing in smart spaces and meeting analysis overlap when it comes to detecting and tracking people non-intrusively in a natural context. In the following sections, some of the current literature on these topics are discussed.

2.1 Smart Spaces

The idea of smart spaces is to use data from sensors to interpret the behavior of humans in the space, such as location, identity and movement, and to allow the humans to interact with the space. Technology advancements are key to realizing smart spaces (D. Surie, Partonia, & Lindgren, 2013). A precondition for spaces to be smart is that people need to be detected and tracked in a non-intrusive way, which is in line with the goal of automating observation of participants in design activities.

Surie et al. (2013) placed a Kinect v1 in a kitchen that was already part of an ongoing project on smart spaces called Kitchen As-A-Pal (Dipak Surie, Lindgren, & Qureshi, 2013). The Kinect was placed on the wall, facing the humans in the kitchen. The human tracker in the supplied Kinect software was used in combination with face recognition to track and recall participants. The system achieved good results (91.75% precision and 66% recall values) for single-occupant setting. Several challenges was identified with multi-occupant settings.

Nakamura (2012) discusses approaches for human sensing in general. How to know what information that should be collected and technologies on how to acquire it. Technologies being highlighted for people detection and tracking are magnetics sensors, image sensors, data glove, beacon/RFID, GPS, gyro sensor.

Teixeira et al. (2010) conducted a survey on the literature of human sensing, focusing on literature for sensing presence, count, location, tracking and identity. The capabilities and limitations of existing sensing solutions were discussed. A unified taxonomy was created and used to structure the solutions. The conclusion of the survey was that future human-sensing systems would most likely consist of massive numbers of binary sensors (usually motion sensors), smaller number of cameras placed at key locations and opportunistic use of sensors on mobile phones.

2.2 Meeting Analysis

Earlier research has done efforts to develop systems that automatically gather data from meetings, or a group setting, in order to analyze the behavior of the participants and how they

interact. A precondition for gathering data about participants are detecting, identifying and tracking them.

Basu et al. (2001) fitted a conference room with sensors and actuators in order to observe and influence human behavior in conversational settings. Auditory and visual data from the room was obtained and analyzed to learn about how people influence each other in a conversational setting. An "influence model" was used to predict how much a person influences other people by evaluating how well the state of one person can contribute to predicting the next state of other people. Five cameras and seven microphones was used. The level of body language was estimated from motion energy in the specified area where the participants were required to sit. Future work consists of further developing the influence model to be able to predict the flow in the conversation.

Stiefelhagen (2002) developed an approach to track the focus of attention of participants in meetings. The focus of attention was assumed to be closely related to their head orientation. A Bayesian approach was used with the head orientation to model at whom a person was looking at. Image data from a panoramic camera placed at the center of the meeting table was used with a neural network, pre-trained with sample images of the participants, to estimate head orientation. The approach identified 73% of the focus of attention correctly. Also other cues was evaluated for predicting focus of attention, such as predicting from who-is-talking. Using who-is-talking in the developed neural net resulted in a 63% accuracy.

McCowan et al. (2005) investigated group actions in meetings by analyzing interactions between the individual participants. Hidden Markov model (HMM) based approaches were used to model the group actions from audiovisual cues of each participant. The audiovisual cues were obtained from a microphone array centered on the meeting table and three RGB cameras placed strategic locations. Combinations of sensing modalities and HMM approaches were tried. The best result was achieved with an audiovisual asynchronous HMM system.

3 Microsoft Kinect

Microsoft launched the first version of the Kinect in November 2011. The Kinect v1 was the first sensor that made 3D sensing technology available at a low price. Computer vision with traditional RGB cameras had been trying for decades to create robust real-time tracking and interpretation of human movements, without success. With the Kinect v1, depth data greatly simplified the task (Z. Zhang, 2012).

The Kinect v1 was initially an attempt from Microsoft to broaden their customer base beyond young adults playing first-person shooting games ("E3," 2009). However, it was soon obvious that the impact of the Kinect Sensor would not only be limited to gaming applications, but also to custom projects and in numerous research fields. Zhang (2012) illustrated the fast impact of the Kinect v1 by investigating the development of Kinect online communities;

"Kinect was launched on 4 November 2010. A month later there were already nine pages containing brief descriptions of approximately 90 projects, and the number of projects posted on KinectHacks.net has grown steadily. Based on my notes, there were 24 pages on 10 February 2011, 55 pages on 2 August 2011, 63 pages on 12 January 2012, and 65 pages on 18 February while I was writing this article. This comment from KinectHacks.net nicely summarizes the enthusiasm about Kinect: ''Every few hours new applications are emerging for the Kinect and creating new phenomenon that is nothing short of revolutionary.''"

3.1 Advantages of Tracking Humans in Depth Data

The task of tracking humans from sensor information, such as camera images, is hard due to a range of factors. The human body has many degrees of freedom and can adopt a multitude of different poses. Further, the anthropometry of individual human bodies are highly variable as well as the bodies being covered with flexible layers of colored skin and clothes. Additionally, the appearance of the scene is variable, with variations in illumination, occlusions and clutter (Plagemann, Ganapathi, Koller, & Thrun, 2010)(Seer, Brändle, & Ratti, 2014).

The depth data from the Kinect is calculated using infrared light that is unaffected by illumination changes and color. In addition, tracking humans in 3D depth data greatly simplifies the task of differentiating between background and foreground as well as allowing the tracking algorithms to use the naturally characteristic 3D shapes of humans in the calculations (Greff, 2012).

3.2 Impact on Research

As mentioned earlier, the Kinect has been widely used in research since its launch in 2011. If a search with the keyword "Kinect" is performed in Elsevier's Scopus, a database for abstracts and citations, 3 844 results appear ("Scopus," n.d.). The research communities within Computer Science have found most use of the Kinect sensor, but also other less apparent communities such as Materials Science, Biochemistry, Genetics and Molecular Biology, Social Sciences and Medicine has benefitted from the sensor (Figure 3.2).

A few examples are hand-gesture recognition (Ren, Yuan, & Zhang, 2011), integrating pointing gestures in brainstorming (Kunz, Alavi, & Sinn, 2014), human-activity recognition (Li, Zhang, & Liu, 2010), body biometrics estimation (Velardo & Dugelay, 2011) and healthcare applications (Bauer, Wasza, Haase, Marosi, & Hornegger, 2011)(Galna et al., 2014)(Torres et al., 2015).

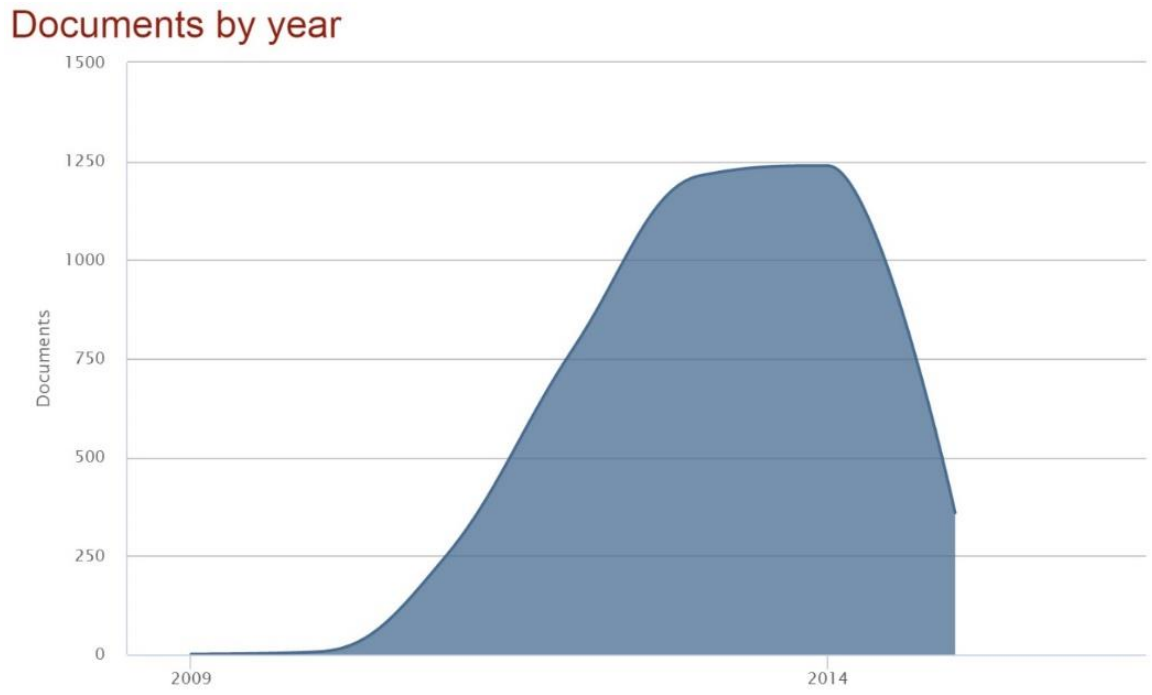


Figure 3.1 Documents in Scopus that appear when the search term "Kinect" is used. Shown by year.

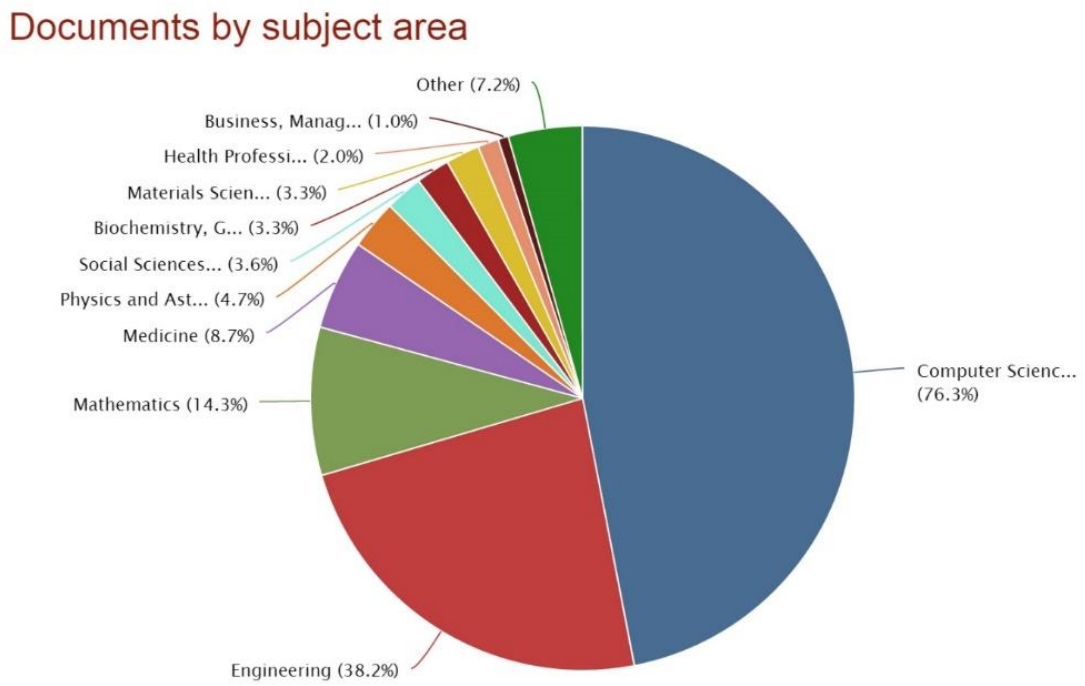


Figure 3.2 The distribution of research fields that has mentioned "Kinect" in publications.

3.3 Kinect v1 vs Kinect v2

Figure 3.3 Kinect v1 on the left, Kinect v2 on the right.

Microsoft launched in July 2014 a new and improved version of the Kinect. The new sensor featured a different depth sensing technology than the previous version, a wider field of view and higher resolution. Table 1 shows a comparison of the specifications of the two sensors.

Component	**V1**	**V2**
Depth Sensing Technology	Structured-light	Time-of-flight
Depth Sensor Sensing Range	1.8 to 4.0 m*	0.5 to 4.5 m
IR Depth Image:		
- Resolution	320 x 240	512 x 424
- Field of View Horizontally	57 degrees	70.6 degrees
- Field of View Vertically	43 degrees	60 degrees
RGB Image Resolution		
- Resolution	640 x 480	1920 x 1080
- Field of View Horizontally	57 degrees	84.1 degrees
- Field of View Vertically	43 degrees	53.8 degrees

Infrared Image	320 x 240**	512 x 424
Audio Stream	16 kHz, 16-bit	48 kHz, 16-bit
Field of View horizontally	57 degrees	70 degrees
Field of View vertically	43 degrees	60 degrees
Minimum Latency	102 ms	20-60 ms

*Table 1: The specifications for the Kinect v1 and v2. Source: ("Kinect for Windows," n.d.). *Recommended sensing distance. Research has shown that max distance with valid data is 0.4 to 6 m(Gonzalez-Jorge et al., 2015). **Was made available first in the SDK 1.6 release.*

Since the launch, the two sensors have already been compared in a few studies. Amon & Fuhrman (2014) evaluated the spatial resolution and accuracy of the face tracking system and concluded the Kinect v2 sensor "features significant improvements to the previous model". Lachat, Macher, Mittet, Landes & Grussenmeyer (2015) concluded that the accuracy and the resolution of the point clouds and the color reproduction has been improved. Further, they concluded that the change of depth sensing technology gives the Kinect v2 sensor greater possibilities to be used in outdoor applications with daylight and even on sunny days.

While the resolution for all data streams and the field of view for the new sensor is improved, current research indicate that it is the change in depth sensing technology that is the most interesting improvement in the new sensor (Gonzalez-Jorge et al., 2015). The change in depth sensing technology was also a precondition for Microsoft to be able to increase the resolution of the depth stream in the Kinect. In the following sections, the two depth sensing technologies will be described and compared.

3.3.1 Structured Light

The Kinect v1 uses a depth sensing technique called *structured light.* The sensor has an infrared projector that emits an infrared laser. The infrared laser is passed through a diffraction grating which turns the laser-beam into a pattern of IR dots, as seen in Figure 3.4. The IR dots covers the scene and are detected by the infrared camera in the sensor. The sensor compares the pattern detected by the infrared camera with the default pattern, and from the distortion between the patterns the sensor is able to calculate the depth map of the scene (Z. Zhang, 2012).

Figure 3.4 The pattern of infrared dots emitted by the Kinect v1. Source: ("Kinect," n.d.)

One disadvantage with this technique is that the amount of depth information extracted from the scene is fixed, independent of the distance to the objects. In practice, this means that if the objects in the scene are far away, a smaller subset of IR dots would reflect back to the IR camera and give information about the depths of the object than if the objects were closer to the sensor. The consequences of this disadvantage are examined in Gonzalez et al. (2015) where it is concluded that the precision of the depth sensing in the Kinect v1 sensor decreases with distance following a second order polynomial. For the Kinect v2 sensor however, the precision is more stable as the distances increase.

Another disadvantage of the structured light technique has been discovered in several projects where multiple Kinect v1 has been applied to the same scene. The Kinect v1s have no way of differencing between the IR dots that is projected by itself and the ones projected by other Kinects, which results in interference between the sensors. A workaround for the interference problem has been developed by the Microsoft Research team and involves making some of the sensors vibrate (Butler et al., 2012). However, this will not be discussed further in this thesis.

3.3.2 Indirect Time-of-Flight

The Kinect v2 uses a depth sensing technique called *indirect time-of-flight* (ToF). This technique is based on calculating the distance to the objects in the scene from the time IR light photons use to travel between the sensor and the objects.

Short infrared light bursts are sent out from three laser diodes (Figure 3.5), covering the scene with short pulses of infrared light. The infrared light is reflected off the objects in the scene and is captured by the infrared camera in the Kinect sensor.

Figure 3.5 The three laser diodes (purple) in the Kinect v2.

In most sensors using the indirect ToF technique, the phase shift between emitted and received signal is measured. The distance to the object from the sensor is then determined by equation (1)

$$d = \frac{\Delta\varphi}{4\pi f} \cdot c \qquad (1)$$

where f is the modulation frequency and c is the speed of light (Kolb, 2009).

However, the way the Kinect v2 measures ToF is different from the most common ToF sensors. The Kinect sensor divides each pixel in half and each half is turned on and off very fast. When the first half is turned on and absorbing photons, the second half is turned off and rejecting all photons. The three laser diodes are being pulsed with the same phase as the first pixel half, so

the sensor knows that if the first pixel half is on, the laser diodes are on. As the infrared light returns to the sensor, the sensor calculates the distance to the objects in the scene by measuring the proportion of light ending up in each pixel half. One issue with this method is that light being reflected off more distant objects in the scene could possibly end up in the next cycle. The sensor solves this by increasing the time the pixel halves are turned on and thereby giving the light more time to be reflected back. Still, increasing the cycle time would lose precision in the closer measurements. So the sensor takes two measurements, where the first measurements is a low resolution (longer cycles) measurement with no ambiguities in distance, and the second measurement is a high precision measurement, eliminating any ambiguities with the results from the first measurement (Butkiewicz, 2014)(Gonzalez-Jorge et al., 2015).

A clear advantage to this technique compared to the structured light technique is that no matter how far the objects in the scene are from the sensor, each pixel in the Kinect v2 sensors gets a depth measurement. Whereas in the Kinect v1 sensor, the depth information retrieved from the possibly limited amount of IR dots has to be shared among pixels, resulting in a higher loss of precision with increasing distance.

3.4 Other advantages with the Kinect v2

Several papers have examined how suitable the Kinect v2 is for outdoors applications in daylight (Gonzalez-Jorge et al., 2015)(Butkiewicz, 2014)(Lachat, Macher, Mittet, Landes, & Grussenmeyer, 2015)(Dutta, 2012)(Jia, Yi, Saniie, & Oruklu, 2012)(González-Jorge, Zancajo, González-Aguilera, & Arias, 2015). While the Kinect v1 did not work very well with ambient lighting, the Kinect v2 is able to give an infrared image independent of ambient lighting. The Kinect v2 achieves this by enabling each pixel to detect if it is over-saturated with incoming ambient light, and if it is, the pixel is reset pixel in the middle of exposure.

The higher resolution also gives interesting possibilities when detection of fine details are needed, such as hand gesture analysis, facial expressions and creating digital models of objects from the depth cloud (Lachat et al., 2015)(Butkiewicz, 2014)(Kunz et al., 2014). At the same time of the launch of the Kinect v2, Microsoft also released an updated version of their Software Development Kit (SDK). In the updated SDK, the facial tracker was greatly improved to take advantage of the possibilities provided by the higher resolution (Figure 3.7, Figure 3.8). Likewise, the Kinect Fusion software created by Microsoft for creating digital models from 3D scans became more accurate with the higher resolution (“Kinect for Windows,” n.d.).

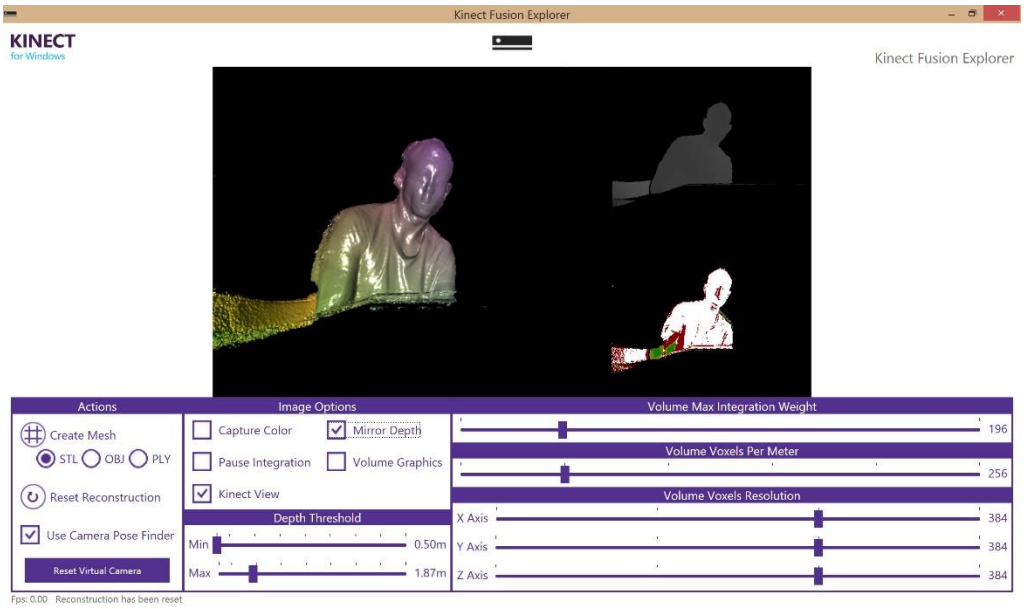


Figure 3.6 Kinect Fusion. 3D models of a scene can be created and exported.

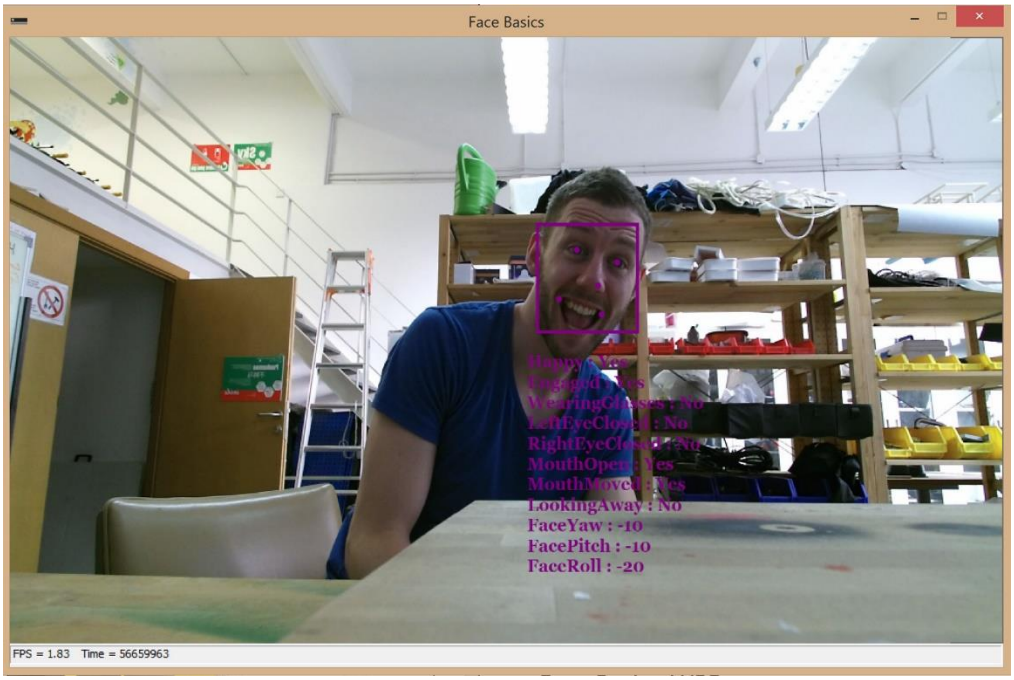


Figure 3.7 Kinect Face Tracking. With the higher resolution, the face tracking has become more precise than before.

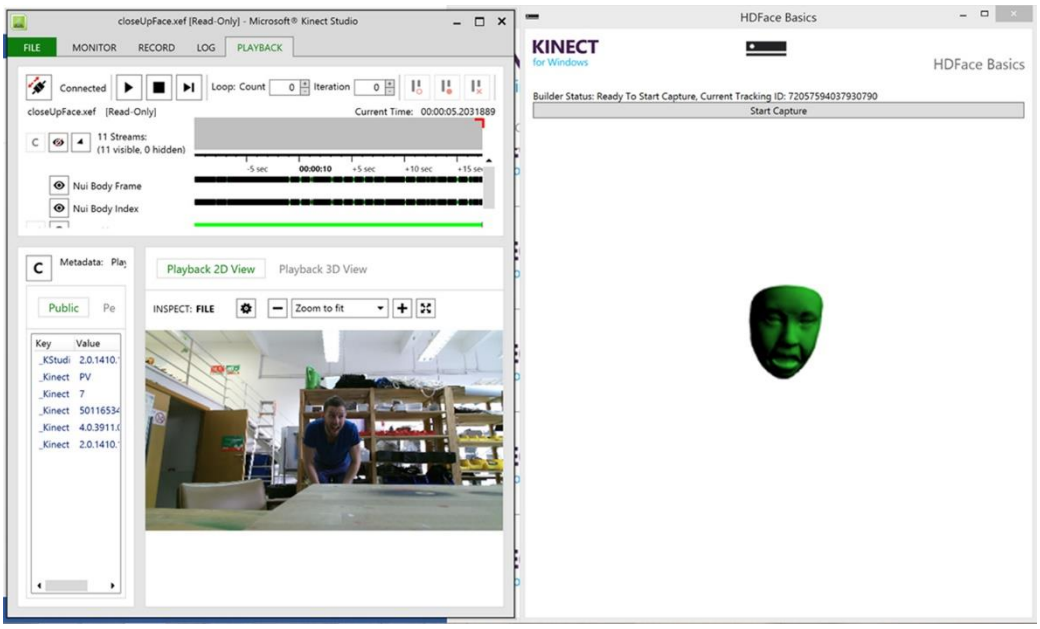


Figure 3.8 Kinect HD Face Mesh. Left: Playback of Kinect recording. Right: The Face API in the new SDK 2.0 allows access to over 1000 facial points in 3D space, which for instance can be used to create precise avatars(Vangos Pterneas, n.d.).

3.5 Disadvantages with the Kinect v2

While there are many advantages with the new Kinect, there are also some disadvantages compared to the old sensor that needs to be mentioned.

According to Microsoft, as a result of the increased amount of data created from the higher resolution, also an increase in bandwidth of the connection to the computer is required. The required increase of bandwidth does that the sensor will only work when connected to a computer through an USB 3.0 Generation 2 connection.

Further, with the Kinect v1, several projects utilized multiple sensors to get a more complete point cloud of the scene. However, as of this moment, the Kinect SDK does not support more than one sensor connected in the same instance of an application (“Kinect for Windows,” n.d.)(“Multiple Kinects,” 2014). In addition, due to the bandwidth requirements, each Kinect sensor needs to have its own USB 3.0 Gen 2 host controller to reserve the bandwidth. According to a discussion thread on GitHub in one of the most popular open-source projects for the Kinect called *OpenKinect*, running multiple Kinect v2s on the same computer has been accomplished using Linux and their open-source driver called libfreenect2(OpenKinect, 2014). Still, this feature seem yet not to be officially implemented in the driver.

Another workaround for the "multiple Kinect issue" has been implemented by both the Kinect community and by Microsoft Research. Each of the Multiple Kinect v2 sensors can be connected to a separate computer and then the data can be synced by streaming it over a local network(Wilson, 2015)(Brekelmans, 2014). Yang, Zhang, Dong, Alelaiwi & Saddik (2015) also used three Kinect v2s simultaneously in their research of further improving the accuracy of the Kinect sensors by trilateration[1].

Another initial disadvantage with the Kinect v2 compared the Kinect v1 is the obviously more limited availability of tutorials and resources online. Even though the SDK 2.0 is very well documented by Microsoft (Microsoft Team, n.d.-a), the Kinect v1 has been extensively discussed online during the past few years and substantial amounts of software and a few open-source drivers has been developed for it. The open-source alternatives for the Kinects are discussed in the section "Open-Source Alternatives to Microsoft Kinect SDK".

Nevertheless, how the data streams are accessed in the Microsoft Kinect API for the two sensors isn't very different. One should be able to port an application written with the Microsoft API for Kinect v1 to a Kinect v2 application in as little as a couple of hours depending on the application.

3.6 Kinect Software Development Kits

In February 2012, Microsoft launched the first Software Development Kit (SDK) for the Kinect (Z. Zhang, 2012). The SDK 1.0 gave the users a more robust and easier to use interface for accessing the data from the Kinect compared to the open-source alternatives. With the Kinect SDK the users could easily develop Kinect applications with Microsoft's Visual Studio in any .NET language, as well as C++, with access to Microsoft's body tracker and audio hardware. How the Microsoft Kinect body tracker works is described in a later section. The open-source alternatives only gave access to the raw data coming from the Kinect so to track bodies a body tracker needed to be implemented. In addition, the SDKs developed by Microsoft are better documented than the open-source alternatives.

The evolution of the Kinect SDKs and Developer Toolkits are summarized in Table 2. Including and after version 1.5, samples and tools for development was put in a separate install called

[1] Trilateration: Each depth measurement is considered as the center of a sphere. The intersection of all spheres with the minimum error is the improved measurement.

Kinect for Windows Developer Toolkit, while the drivers and API still remained in the SDK install.

Date	Kinect SDK and Developer Toolkit version	Added features
5/2/2012	1.0	• Drivers for using Kinect sensor devices on a computer running Windows 7 or Windows 8 developer preview (desktop apps only) • Application Programming Interfaces (APIs) and device interfaces, along with technical documentation • Source Code samples • Support for up to four Kinect Sensors • Skeletal Tracking
5/18/2012	1.5	• **Kinect Studio** • Skeletal Tracking in Near Range • Seated Skeletal Tracking • Joint Orientation • Source Code samples • The Face Tracking SDK • New Supported Languages for Speech Recognition • New Samples
10/4/2012	1.6	• Windows 8 Support • Accelerometer Data APIs • Extended Depth Data Is Now Available • Color Camera Setting APIs • New Coordinate Space Conversion APIs • The Infrared Stream Is Now Exposed in the API • Support for Virtual Machines • New Samples
3/12/2013	1.7	• **Kinect Fusion with Samples: scan 3D objects and create models** • New Kinect Interactions(Press for Selection, Grip and Move for Scrolling)

		• Engagement Model Enhancements • New Samples
9/13/2013	1.8	• Kinect Background Removal • Webserver for Kinect Data Streams • Color Capture and Camera Pose Finder for Kinect Fusion • Updated and New Samples
10/21/2014	2.0	• Kinect for Windows v2 Support • Windows Store Support • Unity Support (Platform for creating 2D and 3D games) • Audio: more precise speech recognition and direction of sounds • **Face APIs** • Kinect for Windows v2 Hand Pointer Gestures Support (greatly improved from v1) • Kinect Fusion: higher resolution and better camera tracking • Kinect Studio greatly improved • **Visual Gesture Builder** • New samples

Table 2: The evolution of Kinect SDKs and Developer Toolkits. Features in bold has been highlighted in the text. Source: ("Kinect for Windows," n.d.)

Kinect Studio, first introduced in SDK 1.5 and later improved, is a very useful tool that has been used extensively in this thesis. It allows the sensing sessions to be recorded and replayed at any given time, which is valuable when developing applications. The data streams are replayed exactly the same way as if there was a live sensing session being carried out.

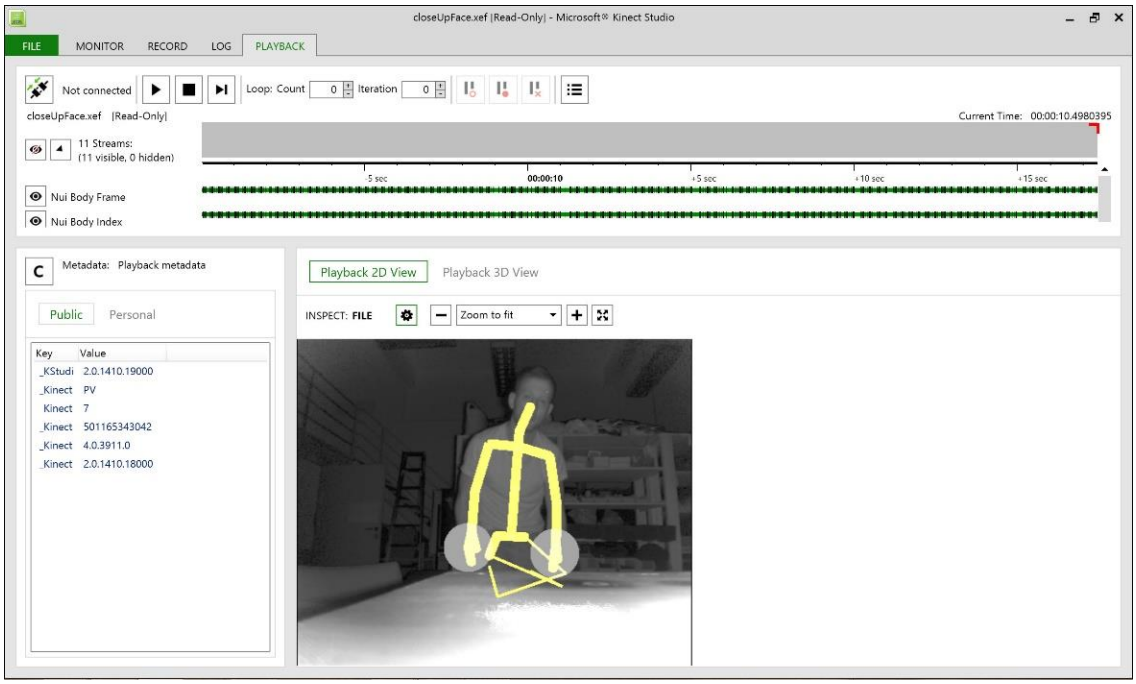


Figure 3.9 Kinect Studio

With the launch of the Microsoft Kinect v2, Microsoft also greatly improved the SDK in SDK 2.0. The higher resolution in the data streams allowed for higher precision in the tracking and tracking of more bodies than the previous SDKs. While the Kinect v1 only was able to track the movement of two bodies, the Kinect v2 can keep track of up to six.

Further, the *Visual Gesture Builder* was introduced, giving the users a graphical user interface for tagging gestures that are used to train machine learning algorithms and create a database of the gestures. The gesture database can then be accessed during runtime to decide if the tracked bodies are performing the gestures. Microsoft provides two different machine learning algorithms, depending of the type of gesture that is to be detected.

If the gesture is a *discrete gesture*, meaning that the user is interested to know the occurrence of a gesture, for example knowing the answer to "is the body currently throwing a punch", then an *Adaptive* Boosting (AdaBoost) machine learning algorithm is used. The AdaBoost algorithm is given frames tagged as either positive or negative to the occurrence of the gesture, which it then uses to train several weak classifiers (Freund & Schapire, 1997). A weak classifier only classifies the input slightly better than pure random, but when many weak classifiers are combined, the results converge towards being well-correlated with true classification. The AdaBoost algorithm learns the optimal weighting of each of these weak classifiers and the combined sum results in a discrete value saying if the gesture is occurring or not.

If the developer wants its application to detect a *continuous gesture*, for example the position of an arm at a certain point in a golf swing, a *Random Forest Regression (RFR)* machine learning algorithm is used. The developer then tags clips instead of single frames where the gesture is being performed, which are used in the training of the RFR. Random Forests are also used in Microsoft Body tracker for the detection of bodies, and is explained in the following section.

3.7 Microsoft Kinect Body Tracker

This section aims at describing the principles of the body tracker developed by the Microsoft Research group. For an even more detailed description, the reader is referred to (J. Shotton et al., 2013), (Kohli & Shotton, 2013) and (Jamie Shotton et al., 2013). Much research of body tracking in depth data from a frontal view has been done throughout the thesis. Summaries of this research are not included in the final thesis, but the research shows that it is the body tracker developed by Microsoft that is currently the state-of-the-art. It is the most precise, robust and fastest body tracker.

Shotton et el. at Microsoft Research published in 2011 the first paper describing the body tracking algorithms used in the Kinect. Since then, the research group has made several improvements to the body tracking model. The main requirements of the body tracker in the Kinect is speed, robustness and flexibility. The Kinect body tracker is required to run several hours without crashing while being able to track users with a great variation in body type. To achieve these requirements, a machine learning algorithm called Randomized Decision Forest was chosen, because this algorithm is efficient and can be parallelized and implemented on a GPU.

3.7.1 Randomized Decision Forest Algorithm

The Randomized Decision Forest algorithm creates several decision trees (Breiman, 2001). A decision tree is a tree where each non-leaf node in the tree contains a simple test that decides the next direction down the tree, for example “is this pixel brighter than the neighborhood pixel”. What test the node should contain is decided using a greedy algorithm that finds the test that creates the best separation between the samples. Each pixel in an image is sent through each decision tree and when the pixel reaches a leaf node, the pixel is attributed with a probability distribution of the possible classifications already stored in the leaf node. The previous training of the decision tree decides the probability distribution stored in the leaf nodes

by using the training samples to calculate a probability distribution of a pixel with a certain classification reaching that exact leaf node.

As the possible tests at each node and the number of training samples become large, building an optimal tree becomes too computationally demanding. Therefore, multiple randomized trees are created. Each tree is trained with a limited amount of randomly chosen samples and a limited amount of possible tests at the nodes (Lepetit, Lagger, & Fua, 2005). After the pixel has traversed through all decision trees in the decision forest, the probability distributions attributed to the pixel from all the decisions trees are averaged together and creates the hypothesis for the classification.

The Microsoft Kinect body tracker uses comparison of depths of pixels instead of comparing pixel intensities (J. Shotton et al., 2013). At each non-leaf node the depth of the current pixel is compared to the depth of a pixel with a certain offset from the current pixel. The node tests at a given pixel **u** for the Microsoft body tracker can be described as

$$f(\mathbf{u}|\varphi) = z\left(\mathbf{u} + \frac{\boldsymbol{\delta}_1}{z(\mathbf{u})}\right) - z\left(\mathbf{u} + \frac{\boldsymbol{\delta}_2}{z(\mathbf{u})}\right) \tag{2}$$

where $\varphi(\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$ describe the 2D pixel offset and function $z(\mathbf{u})$ looks up the depth at pixel **u.** Each body part will have different distributions of the probability of what depth their surrounding pixels will have. For example, a pixel where there is a big difference in depth between the current pixel and the pixel above the current pixel would get a higher positive response to being a head pixel than for instance a torso pixel.

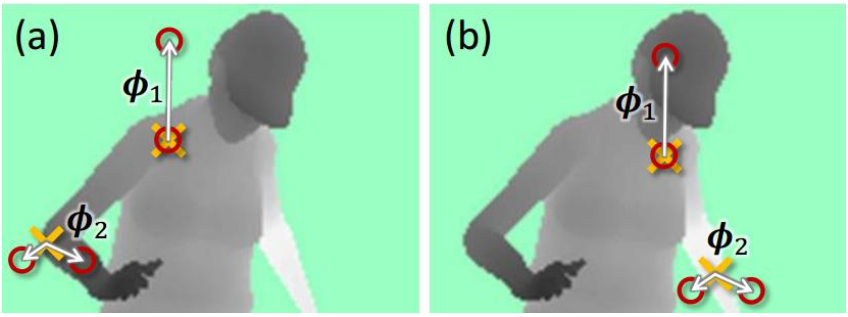


Figure 3.10 Test in leaf node. The depths of the current pixel and pixels at predefined offsets are compared as tests in the decision tree. Source: Budiu, Shotton, Murray & Finocchio (2011).

3.7.2 Leaf Node Prediction Models

When the first Kinect was shipped, a Body Part Classification (BPC) algorithm was used to predict to what body part the pixel should be assigned. In BPC, a histogram representing the probability distribution over the body part labels that the pixel should be assigned was learned at each leaf node. The histograms from all the pixels were then clustered together to give reliable hypotheses for the location of each joint. One big downside of the BPC algorithm was that no information could be obtained about the joints when the surrounding body parts were occluded.

To meet the challenge of occlusion, the Microsoft research group developed an offset joint regression approach (OJR) that was implemented in the leaf nodes (Girshick, Shotton, Kohli, Criminisi, & Fitzgibbon, n.d.). In the OJR, each leaf node contains a distribution of the relative 3D offset from the projected pixel in camera space coordinates to each body joint. These distributions are matched with 3D relative vote vectors that are created from the offsets of the clustered pixels from training the algorithm.

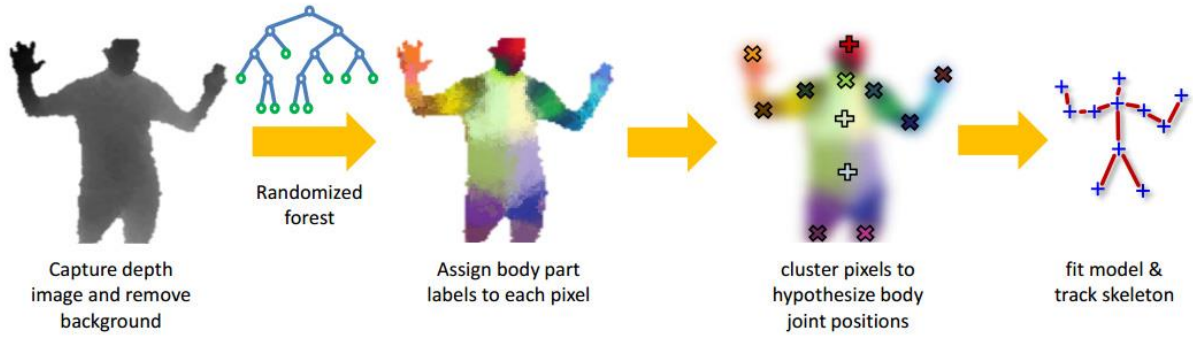


Figure 3.11 The pipeline of the Microsoft body tracker. Source: Kohli & Shotton (2013).

Even though this approach worked well, there was still no utilization of prior knowledge of the bodies tracked, and in addition, all joints were detected independently so the model did not take into account any information about the relative position between joints. These potential areas of improvement was addressed in Min Sun, Kohli & Shotton (2012) where a latent variable was introduced in the model to encode some global property of the image, for example the height of the tracked body.

A further potential area of improvement was to include information about the kinematic constraints such as limb length into the model. In Sharp (2012) the work of the Microsoft research group of fitting an articulated skeleton model to the observed data is described. In 3D

space, the translation between the joints are fixed, representing fixed limb lengths, while the rotation in the joints are parameterized.

3.7.3 Creating Training Data

One of the challenges with choosing a machine learning approach is that for the tracking to be robust, the algorithms need to be trained with training sets containing large varieties of poses. Unfortunately, possible human poses grow exponentially with the number of articulated joints.

Sufficient training sets did not exist, especially not with depth data, so Microsoft had to create their own (J. Shotton et al., 2013). The Microsoft Research team created a large training set with human poses by using marker-based motion capture of real human actors. 500 000 frames was recorded in a few hundred sequences of their core entertainment scenarios (dancing, kicking, navigating menus etc). From this set, a subset of 100 000 of the most dissimilar frames was selected. To take the great variations in human shape and appearance into account, 15 3D models of varied base characters in terms of gender, age, height and weight were created. These 3D models were randomly paired with a pose from the recorded frames, as well as randomly assigned a rotation & translation, mesh models of hair & clothing, further variation in weight & height, camera position & orientation and camera noise to make the final rendered training set as realistic as possible. Training the Randomized Decision Forest also required a lot of resources. According to Shotton et al. (2013), training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster.

3.8 Open-Source Alternatives to Microsoft Kinect SDK

The only real current alternative to the Microsoft Kinect SDK, for both Kinect v1 and v2, is the *libfreenect* and *libfreenect2* drivers developed by the open community *OpenKinect*. The community boasts having over 2000 members contributing to the project, although it seems the activity has been decreasing the last years (Figure 3.12, Figure 3.13). The main advantage by using the *libfreenect* software for the Kinect v1 sensor is support for Linux and OS X as well as bindings and extensions for additional languages such as Java and Python. Also, work is under development for exposing the API to MatLab, LabView and more ("OpenKinect Project," n.d.).

For the Kinect v2 sensor, the *libfreenect2* driver is an alternative, although it is far from as developed as the driver for the Kinect v1 sensor. The only advantage of the *libfreenect2* software is the possibility of accessing the Kinect v2s data streams in Linux and OS X. Additionally, the *libfreenect2* driver only supports the RGB, IR and depth data streams. Audio

and firmware updates are not available. Both drivers, *libfreenect2* and *libfreenect*, only gives access to the data streams and don't include any tracking whatsoever or support for Kinect tools, such as Kinect Fusion and Visual Gesture Builder.

Previously, another alternative for the Kinect v1 sensor called *OpenNI* (Open Natural Interaction) existed. The *OpenNI* framework and the middleware for body tracking called *NITE* was developed by *PrimeSense*, the same company that developed the depth camera technology in the first Kinect ("PrimeSense Supplies 3-D-Sensing Technology to 'Project Natal' for Xbox 360," 2010). Unfortunately, after the company was acquired by Apple in April 2014, the company stopped maintaining and developing the software.

Figure 3.12 libfreenect on GitHub. Contributions to the master-branch on GitHub for the libfreenect driver for Kinect v1.

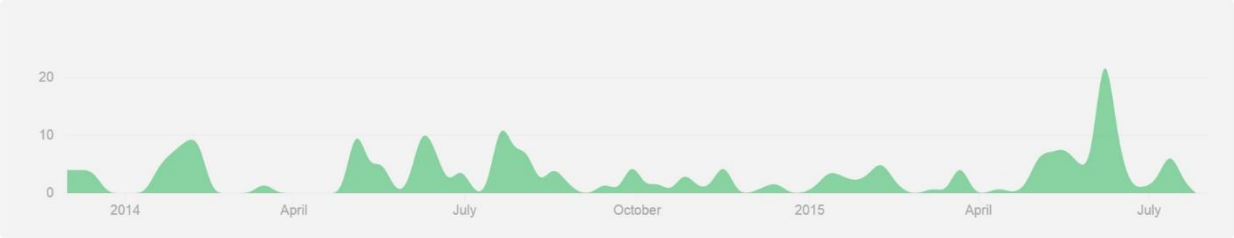


Figure 3.13 libfreenect2 on GitHub. Contributions to the master-branch on GitHub for the libfreenect2 driver for Kinect v2.

4 Placing the Kinect sensor

The Kinect sensor, and the state-of-the-art body tracking software developed by Microsoft, are meant to be used with the human bodies standing up straight directly in front of the sensor. Also, so far, Microsoft does not support more than one of the Kinect v2 being utilized in the same application. Due to these limitations, the placement of the sensor needed to be done with great care.

Occlusion is the main challenge for body tracking in depth data. Occlusion is described as two types: normal occlusion and self-occlusion (Asteriadis, Chatzitofis, Zarpalas, Alexiadis, & Daras, 2013). Normal occlusion is when a body or parts of the body are being hidden from the sensor by either an object or another body, as the image on the left in Figure 4.1. Self-occlusion is the case when a body is oriented in a way relative to the camera so parts of body is hidden behind other parts of the body, as in the image on the right in Figure 4.1. Intuitively, a way of avoiding this problem is by placing the sensor high up with a clear view of the bodies. There are rarely cases where objects or other bodies are positioned on top of the body that we want to track.

Figure 4.1 Occlusion: Left: Normal occlusion. Right: Self-occlusion. The tracking results of Microsoft Body tracker has challenges matching the ground truth.

The Microsoft body tracker are also *inferring* body landmarks, joints, that aren't directly detected. To evaluate the performance of the Microsoft tracker at different heights, a diagnostics tool was developed to calculate the amount of tracked and inferred joints during a sensing session. The sensor was positioned at 10, -30 and -60 degree angle with the floor and a rehearsed routine was performed to minimize the differences in the scene between the angles. The routine started with standing up straight facing the sensor, then the hands were lifted straight out so the body formed a T-shape and down again. After the hands returned to their neutral straight position, a 90 degree turn to the right was performed so the sensor only could see the body in profile. The 90 degree turn was performed until the body returned to its starting position, facing the sensor. The tracking success for each sensor-position was averaged over three sessions for each position. The sensing sessions, 9 in total, lasted 12.01 ± 2.10 seconds.

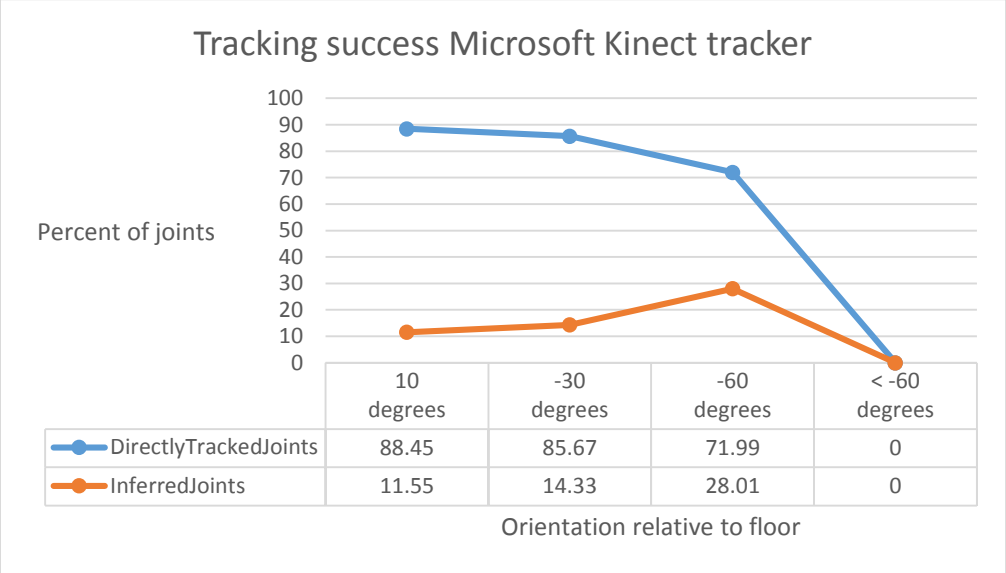


Figure 4.2 Tracking success of the Microsoft Kinect tracker at different angles.

At first sight the results from the tracking with the Microsoft tracker seem very impressive. However, a more thorough inspection of the sensing sessions shows that even though a joint is reported as tracked or inferred it still may vary a lot from the ground truth. Figure 4.3 shows examples of this. In addition, at an angle of -60 degrees it was difficult for the Microsoft tracker to find the body in the frame. To initiate the tracking the test person needed to lean backwards to make the angle between the body and the tracker closer to the normal facing angle. At angles below -60 degrees the tracker wasn't able to recognize a body in the frame, which means that

no information about the body is available through the API (joint positions, audio tracking, face tracking etc).

Figure 4.3 Microsoft Body Tracker. Even though the Microsoft Body tracker achieves a tracking result, the tracking may be far from the ground truth. Left: Tracking at -30 degrees. Middle and Right: Tracking at -60 degrees

4.1 Possible Observation Setups

In a real life work setting, you cannot rely on that all people in the sensing area are facing the sensor at all times. An important requirement for the sensing approach for design contexts are that the people should be allowed to work as freely and natural as possible. The sensing should not be noticeable by the subjects. To achieve this, a customized sensing approach needed to be developed. Three main approaches were considered.

4.1.1 Alternative 1: One Kinect v2 and Using the Microsoft Tracker

The Microsoft body tracker available through the Kinect SDK 2.0 is the state-of-the-art in non-intrusive people tracking. The disadvantage is that the sensor needs an unobstructed frontal view of the bodies in the scene. One alternative was to place a Kinect v2 sensor high up on a wall slanting down facing the people. If the sensor wasn't placed too high Microsoft body tracker could be used Figure 4.4.

The major disadvantage with this setup is that the activities and the work setup for the participants in the scene need to be carefully controlled, so the participants always are facing the sensor and aren't occluded.

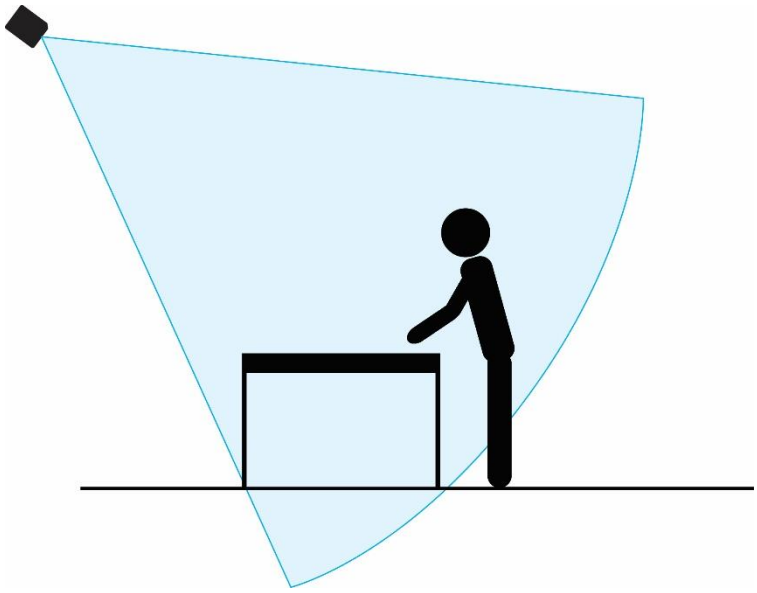


Figure 4.4 Setup with one Kinect v2 slanted down at the participants.

4.1.2 Alternative 2: Two Kinect v1

While using several Kinect v2 in the same application isn't supported, using several Kinect v1 is. By using two Kinect 1 sensors placed as in Figure 4.5, the sensing area would be more versatile than in alternative 1 because it wouldn't be as sensitive to occlusion.

On the other hand, the Kinect v1 has worse resolution and a narrower field of view than the Kinect v2. Further, the depth measurements from the Kinect v1 are less accurate than the Kinect v2, as will be thoroughly discussed in a later section. If this setup was chosen, the body tracking approach would have been to combine the joint-locations from the old Kinect SDKs, which is not as precise as the SDK 2.0. Also, only information about two bodies per frame is available with the old SDK. An alternative would be to use open-source projects to merge the depth clouds and then develop a tracking approach in the merged cloud. The extra work of calibrating and merging depth clouds, together with the fact that the old Kinects weren't accessible to the project before late, made this alternative not a good option.

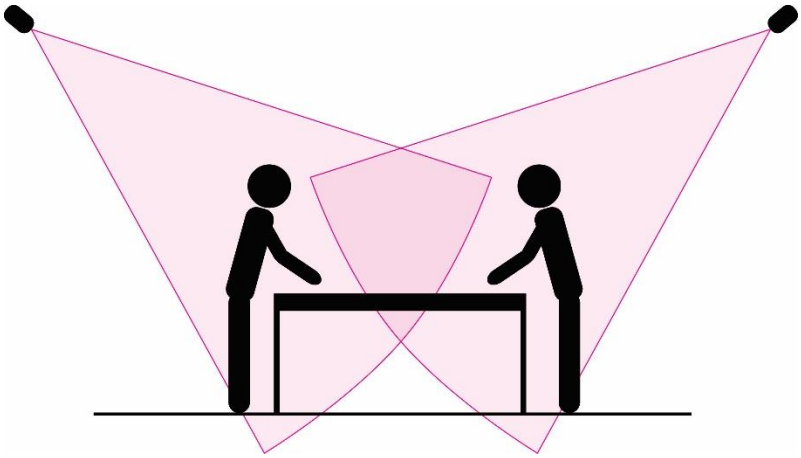


Figure 4.5 Setup with two Kinect v1s mounted high to minimize occlusion.

4.1.3 Alternative 3: One Kinect v2 with Top-View

The approach that was chosen was to position a Kinect v2 high up, facing the floor perpendicularly. The wider field of view and the higher resolution in the Kinect v2 made it possible to place the sensor high above the floor and still get a usable sensing area and accuracy in the depth measurements. With a top-view approach, many of the challenges with occlusion was also naturally solved.

Disadvantages with this alternative are that the Microsoft body tracker could not be used. It is not trained with training images from a top-view position and cannot detect bodies from this position. A body-tracker had to be written totally from scratch, or partially with the help of existing open-source libraries. Another challenge is to be able to position the sensor high enough. The setup is depending on the ceilings being high enough and the ceiling being accessible for mounting.

Alternative 3 was chosen because it was considered to be the setup with most potential and it was the alternative where the newest technology would be used.

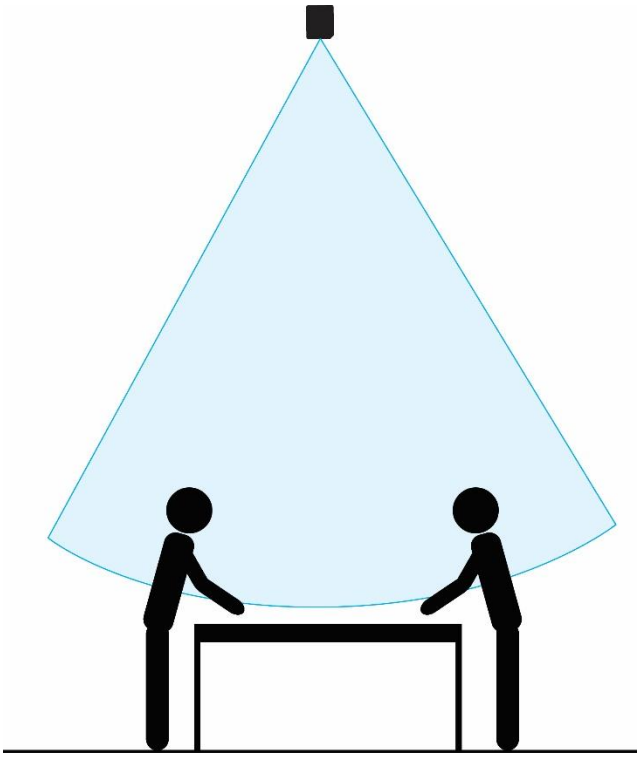


Figure 4.6 Setup with one Kinect v2 mounted in the ceiling.

5 Existing Research: Human Tracking in Depth Data from Top-View

A precondition for automating design observation and the task of interpreting human behavior is to track the humans in the scene. The following sections will describe existing research on how this can be done for the chosen top-view setup in depth data. The chapter will start by describing approaches for background subtraction, then detection and tracking approaches. Strengths and weaknesses will be discussed. Finally, the most relevant open-source computer vision libraries are discussed.

5.1 Background Subtraction in Depth Data

Background subtraction is a crucial step for detecting humans in an image frame. It is usually one of the earliest stages in the pipeline, and if done successfully, greatly simplifies the rest of the detection and tracking process. As indicated by the name, background subtraction is the process of differentiating between the objects you want to track, in most cases humans, and the rest of the scene. Depth data gives several advantages for background subtraction. Subtraction with traditional RGB camera relied on detecting differences between the color intensities of the foreground and background pixels. An approach that is very sensitive to illumination changes and that has to handle great varieties in color and texture in the scene. With the depth data available, background subtraction can rely on the more naturally stable 3D difference between objects and background, which has shown itself to be more reliable (Fernandez-Sanchez, Diaz, & Ros, 2013). Still, color data from RGB cameras don't contain as much noise as the depth measurements, typically at object boundaries, and lacking depth data due to cast shadows or badly reflecting materials such as mirrors and pitch black textures. In this section, some of the state of the art background subtraction approaches are presented. Some algorithms are traditionally used in RGB images but adapted for depth data, some are purely developed for depth data and some use both RGB and depth data.

Stone et al. (2011) modelled the depth map of the background in a training phase before the actual detection phase. The approach is called *minimal background*. After the training phase, all depths that was shallower than the background was considered foreground pixels. This approach works well when all of the objects that should be detected always appear in front of a static background. However, for dynamic backgrounds the subtraction will become noisy or depth camouflage might happen. Depth camouflage is the case when an object in the scene is modelled as background and then moved further away from the sensor. If then a person, that

should be detected, moves to the previous position of the background object, the person will be subtracted from the depth map. Zhou & Aggerwal (2001) and Kepski & Kwolek (2004) also did a similar approach, but created the background model from *median filtered* depth maps and updated the background model throughout the detection session.

Tseng et al. (2014) performed *graph-based-segmentation* to subtract the background from the foreground. In graph-based-segmentation the depth map is divided into candidate regions by merging pixels that are close to each other in Euclidean distance. Further, iterations are done so regions that are closer to each other than the internal distance between the pixels inside the respective regions are merged. Graph-based segmentation is an approach adapted from background subtraction in RGB data. The approach is in general computationally intensive and the candidate regions still needs to be validated for the foreground to be established.

Greff (2012) compared several approaches for background subtraction in depth data. It was concluded that the best performing approaches was the already mentioned *minimal background* approach and an approach called the codebook model. The codebook model is an algorithm previously developed for RGB images but adapted to also take advantage of depth data (Fernandez-Sanchez et al., 2013). In the codebook model algorithm, each pixel is given a codebook with code-words (Kim, Chalidabhongse, Harwood, & Davis, 2005). A code-word is a data structure that contains information about the pixel, such as intensity, frequency of use and depth. After a training phase, each pixel in the new frames are compared to its respective pixel in the codebook from the training phase. If the difference in intensity, depth and so on is big enough, the pixel is classified as a foreground pixel. The codebook approach works better than the minimal background approach when there is a periodic dynamic background. Its disadvantages are higher computational costs and higher complexity.

Nguyen et al. (2015) proposed an approach using both depth and color data. Two classifiers were used, one based on depth and one on color data. The weighting between the classifiers was adapted for each pixel, depending on if it was close to an edge and the color gradient at its location. Near the edges of an object, the color based classifier had greater influence. This was an attempt to reduce the error from noisy depth measurements at the borders of an object. Where the gradients in the area surrounding the pixel was low, the depth classifier was emphasized. The method handles challenging situations with inaccurate depth measurements well. Still, at a computational cost.

5.2 Detection and Tracking in Top-View

As explained earlier, choosing a detection approach where the sensor is placed at a position high above ground and facing down at the participants has several advantages. Some of the most obvious application areas for a top-view detection approach has been surveillance and crowd analysis (Tseng, Liu, Hsiao, Huang, & Fu, 2014). This section will map out the state of the art for detection approaches with a top-view placement of depth sensors.

Tseng et al. (2014) mounted several Asus Xtion Pro[2] cameras with a 3.5m height and stitched the depth data together. A graph-based-segmentation approach was used to subtract the humans from the background. The subtracted blobs were compared to a hemi ellipsoidal model of the human head. If the error between the hemi ellipsoidal model and the blob is below a certain threshold the blob is considered a candidate head. The candidate head was further verified by creating a geodesic distance map from the center point of the head (explained in section *"Geodesic Distance Map"*). The geodesic distance map was used with a detector based on Histogram of Oriented Gradients – Comparison of Granules, trained to detect the shoulders (Dalal & Triggs, 2005). To increase the precision of the tracking and avoid tracking drift, 3D diffusion distance was used to characterize each detection and compare the detections between frames. The 3D diffusion distance algorithm characterizes a detection by dividing the area around the center of the head into bins in a histogram. The similarity between the histograms in consecutive frames are compared. The algorithm outperformed several other state-of-the-art methods. A challenge with this detection approach is missing data in the raw depth image, making the blobs from the graph-based-segmentation fragmented. Further, when creating geodesic distance maps from a top-view, the minimum threshold between the data points in the map needs to be high for the map to be able to continue over the sudden changes in depth around the contours of the head. A high threshold will lead to the geodesic map bleeding over into surrounding objects, if they are close enough to the head and not subtracted away in the pre-processing phase. Lastly, the detection approach of Tseng et al. (2014) is only suitable for detecting humans when walking or standing up straight, not when doing movements such as leaning forward or bending down.

Oosterhout et al. (2011) used connected component[3] to separate the subtracted foreground into blobs. The size of the blobs were evaluated and the blobs with an inappropriate size were

[2] https://www.asus.com/us/Multimedia/Xtion_PRO_LIVE/

[3] Wu, Otoo & Suzuki (2008)

discarded. The remaining blobs were then searched using a template matching approach. The template consisted of two spheres with the same center point and different radius. To be considered a head, the candidate depth points needed to fit between the boundaries of the two spheres. The size of the spheres were adjusted to anatomically plausible values. After detection was performed, the heads were tracked using Kalman filters on the projection of the heads in the 2D plane (Kalman, 1960). The method of Oosterhout et al. works well even for crowded situations compared to other state-of-the-art approaches.

Tian et al. (2013) chose to subtract the background by a simple depth threshold – all data below a certain height were discarded. The remaining data was normalized with gray values between 0 and 255 and used with a Histogram of Oriented Gradients (HOG) feature descriptor. The HOG feature descriptor counts the orientation of the gradients in localized portions of the image. The distribution of the gradients is the descriptor of the respective local region (Dalal & Triggs, 2005). Further, the features were classified by linear *Support Vector Machine (SVM).* The linear SVM was trained with images of positive and negative samples and a model that assign new input to one of two categories were built.

Zhu & Wong (2013) developed a sliding window approach based on Haar-like features (explained in section "Haar-like Features*"*). The Haar-like features of Zhu & Wong was based on assumptions about the area around the head of a human – empty space in front and behind the head, a certain height difference on the sides down to the shoulders and empty space besides the shoulders. The Haar-like features were used in the AdaBoost algorithm (explained earlier) with their own training set. After detection of the human, the tracking was made more precise and stable using Kalman filters. Challenges with this approach occurs when the assumptions about the area around the head is disrupted. This happens when the person is leaning forward or is standing close to something. Another issue with this approach is that the detection window with the Haar-like feature that was used wasn't rotationally invariant, therefore, with the setup presented by Zhu & Wang the detection will become imprecise when a person isn't facing exactly in the Y- or X-direction.

Seer (2014) mounted multiple Kinect v1s in the ceiling of a busy corridor at MIT. The aim of the setup was to track pedestrian behavior. Foreground subtraction was performed partly the same way as Zhu & Wong, with a heuristic cutoff at a certain height. Agglomerative clustering was then performed to form groups of points belonging to individual people (Duda, Hart, & Stork, 2000). A straight forward approach was chosen to identify the heads of the people – by

deciding the head location to be the location of the point with the 95th percentile height of the cluster. The detection rate was reported to be high (94-96%).

Zhang et al. (2012) assumed the head to always be the highest point on the body, so detecting the heads of the people in the frame equaled finding the suitable local minimum regions. To explain the algorithm, an analogy of "raindrops" flowing to the local minimum regions was used. Uniformly distributed raindrops were generated over the frame. The raindrops would then land and flow to the shallowest point. A measurement function was created to filter out the holes that weren't deep enough or that didn't contain enough water. The remaining holes were considered to be heads. The algorithm was reported to perform well compared to other state-of-the-art algorithms, with a 99% accuracy on the created dataset. Still, in situations where the background subtraction is imprecise or when there are objects in the frame that is located higher than the heads, challenges will occur in the detection.

6 Open-source Computer Vision Libraries

Using open-source computer vision libraries might reduce the time needed to develop a custom top-view body tracker. Several were considered throughout the process. None of the libraries that was found had any modules that could directly be used to detect humans from a top-view. Many libraries had implementations for more commonly requested functionalities, such as face detection, blob detection and image filters. Importantly, the libraries should be available in C++ or C# as the API and the examples for the Kinect v2 are only available in these languages.

OpenCV[4] is one of the most popular open-source libraries, and is the most comprehensive and fastest library I considered. The core of OpenCV is written in C++ and has wrappers for Python and Java. The library did not out of the box provide any useful functionality. The modules for processing of depth data are limited. The depth data could have been represented as for instance grayscale values and processed with traditional computer vision techniques, but this didn't seem like a good approach. Additionally, I did not have any experience with C++ before the thesis and wanted to avoid the overhead from learning that. One possible reason for choosing C++ would be because of performance benefits. However, research showed that C# matches the speed of C++ (Qwertie, n.d.).

AForge.NET is a computer vision framework in C#. The framework has classes to directly access the depth data from the Kinect v1. Unfortunately, no support exists for Kinect v2, and the examples provided by Microsoft do the same thing. In addition, no further functionality for processing depth data seems to exist.

Other open-source libraries considered was MATLAB Image Acquisition Toolbox and Point Cloud Library[5]. MATLAB has support for the Kinect v1, but not the Kinect v2. The Point Cloud Library may have had some useful noise removal methods, but is written in C++ and wasn't used for the same reasons as OpenCV.

[4] http://www.opencv.org/

[5] http://www.pointclouds.org/

7 Existing Research: Energy from Human Movement

Energy expenditure is traditionally estimated through heart-rate monitors, accelerometers or metabolic approaches where the O2-consumption is measured and from which the energy expenditure is derived (Nathan, Huynh, Rubenson, & Rosenberg, 2015). All of these methods are intrusive. In addition, the metabolic methods and using heart-rate data are unable to give real-time responsiveness to the energy expenditure, so locating the source of the change in energy expenditure is difficult. Estimating energy expenditure from movement using an external sensor, such as the Kinect, has the potential of giving real-time responsiveness at the same time as not being intrusive (Nathan et al., 2015).

Still, some unavoidable limitations exist when energy expenditure is estimated with an external sensor. The external sensor can only measure mechanical energy expenditure, not internal expenditure such as thermoregulation, digestion and energy expenditure from maintaining basic body functions(Van de Walle et al., 2012). Further, isometric contractions, co-contractions or work against gravity cannot be accounted for(Williams, 1985). The mechanical energy expenditure is the aggregated kinetic and potential energy spent by the human body.

Van de Walle et al. (2012) compared three approaches of measuring energy expenditure from mechanical movement in walking, and compared with control measurements from a metabolic approach. The three approaches were:

1. Center of mass (CoM): analysis of changes in center of mass for the whole body and changes in body segments relative to CoM.
2. Sum of segmental energies (SSE): analysis of energy changes of moving body segments. SSE was calculated by determining total energy per segment and summing them.
3. Integrated joint power: integration of power around all joints as obtained from the VICON Plug-in-gait model.

The aim of the study was to evaluate the usefulness of estimating energy expenditure from mechanical movement, to be able to discriminate between pathological and typical gait as indications of cerebral palsy. In order to achieve this, high sensitivity in the measurement approaches was sought for. A VICON camera system[6] with markers on the body was used to capture the mechanical movement and two embedded force plates was used to capture forces.

[6] http://www.vicon.com/

The conclusion of the study was that approach 1 underestimated the total energy expenditure and showed low correlation to O_2-cost, because negative work done by the muscles isn't considered in this approach. Approach 2 and 3 showed low to moderate correlation, but higher correlation wasn't expected since mechanical approaches cannot account for all sources of energy expenditure. Approach 3 is the recommended approach if collection of bilateral kinetics is possible, if not approach 2 is a valuable alternative.

Nathan et al. (2015) demonstrated that energy expenditure during exercise, using the Kinect v1 sensor as a motion capture system, can be estimated from segmental mechanical work. The motion data from the Microsoft SDK of two Kinect v1 sensors was combined to capture the mechanical work in several steady-state exercises. As in the CoM-approach in Van der Walle et al. (2012), they defined an energy model including both external work (work that moves the CoM of the whole body) and internal work (work that moves body segments relative to CoM). In addition, they improved the estimations of body segment properties, such as segment mass, length and CoM-position, by using the Zatsiorsky-Seluyanov's equations of de Leva(de Leva, 1996). The variables from the mechanical work model were used as feature vectors in several predictive models to predict metabolic energy cost, where a Gaussian Process Regression model gave the best results. The conducted experiments consisted of steady-state arm swings, standing jumps, body-weight squats and jumping jacks. The exercises was repeated for a minimum of four minutes with a resting period in between. The energy expenditure was control-measured with a metabolic system. The study concluded that the energy expenditure can be predicted with the presented model from mechanical work.

Liu et al. (2012) estimated energy consumption by calculating external work of body parts relative to the environment from movement, also using a Kinect v1 sensor. An average human model was defined with 175 cm height and 65 kg weight. The mass of each body part was estimated to a certain percentage of the total body weight. For instance, the head was considered to consist of 23.1% of the total body weight. From a frontal view, the participants were tracked while dancing. The total energy consumption was estimated by calculating change in kinetic and potential energy for each body part between consecutive frames.

8 Proof of Concept: Tracking in Design Activities in Depth Data from Top-View

From research, it turned out that no tracking software was readily available for tracking people in depth data from top-view. Further, of the open-source computer vision libraries that was tested, most were either slow or would lead to unnecessary extra work from adapting the libraries to the specific context. Extra work that wouldn't justify the advantage of features already being implemented.

A few requirements of the tracking approach should be:

1. *Should be non-intrusive*
2. *Be able to instantly detect all bodies in the frame, independent of the bodies' orientation or pose.*
3. *Detection of the whole body is necessary, so a precise heat map of movement can be calculated.*
4. *Center of Mass of body parts should be estimated from which mechanical energy-use can be derived.*

As the setup with a Kinect v2 position with top-view was chosen, ideas for the customized tracking application were obtained from existing research on top-view people detection and detection in depth data in general, but also from traditional computer vision approaches for RGB images.

8.1 Detection Pipeline

The overall detection pipeline of the tracking application is shown in Figure 8.1. The following sections will thoroughly explain each step.

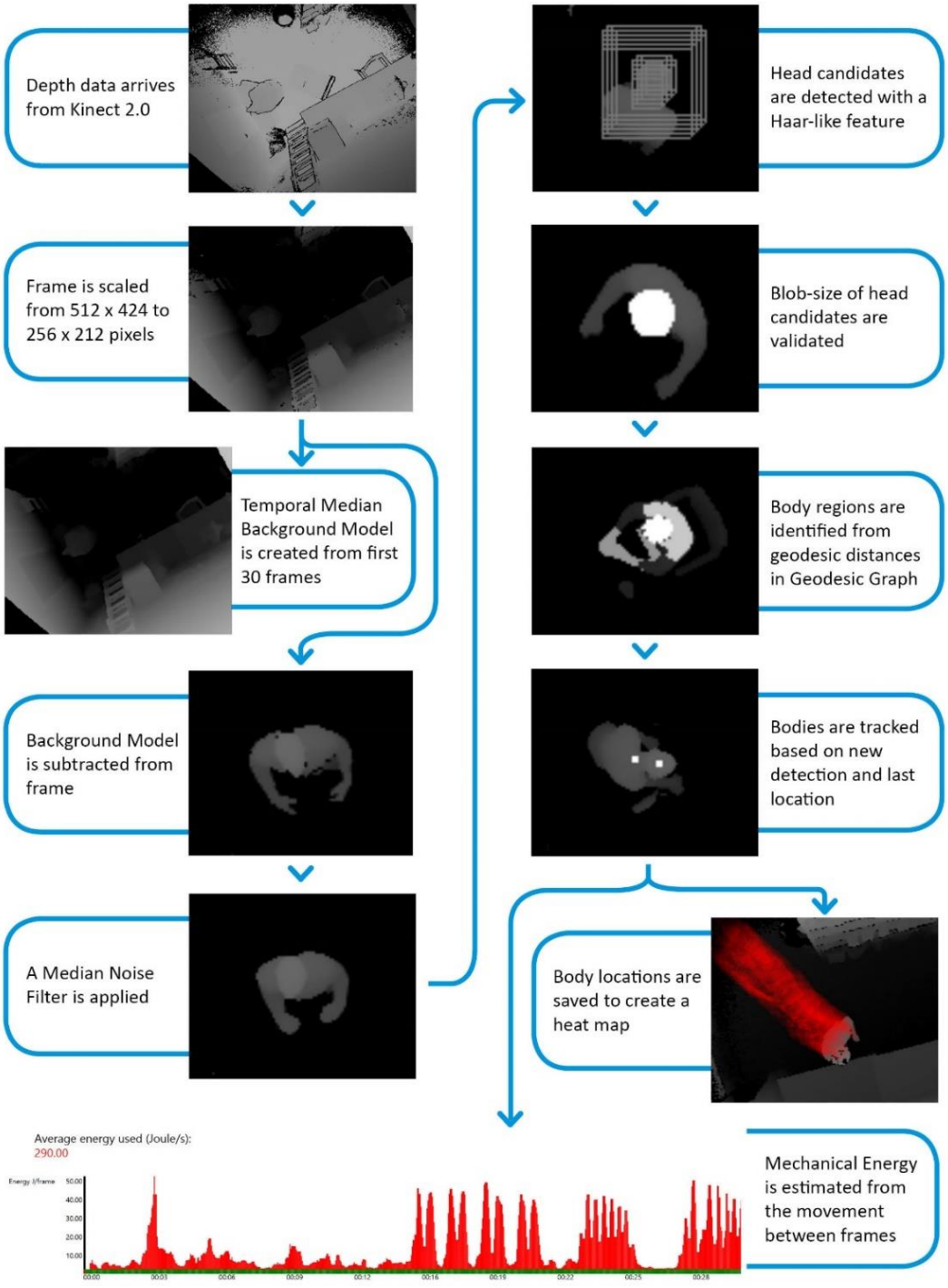


Figure 8.1 Pipeline for the tracking approach.

8.2 Preprocessing: Scaling

Each depth frame from the Kinect sensor is scaled from 512x424 pixels to 256x212 pixels to reduce computational load. A scaling-algorithm was developed and customized to the depth data from the Kinect sensor. Because depth data contains a lot of noise, the algorithm always tries to preserve as much valid data as possible during scaling. More precisely, the algorithm extracts depth information from a region of four pixels into one scaled pixels. The average X-, Y- and Z-coordinates of the four pixels is calculated and used in the resulting pixel. If a pixel in the region does not contain valid depth information, the pixel is excluded from the calculations. The algorithm is shown in Algorithm 1.

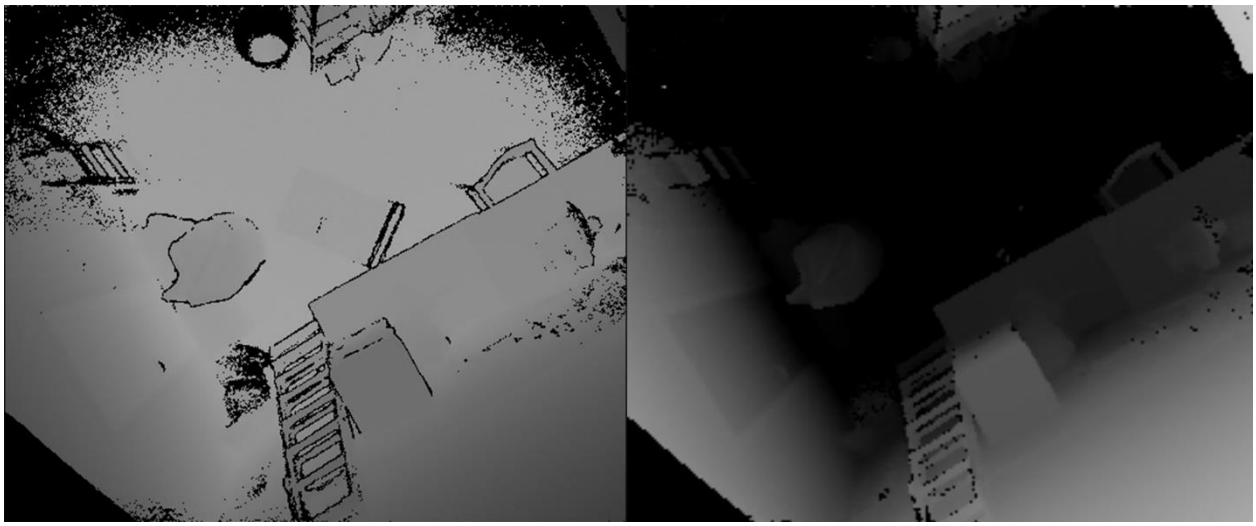


Figure 8.2 Scaling. The depth frame before (left) and after (right) scaling. The grayscale gradient to illustrate the depth is also reversed.

Algorithm 1 Scaling the frame

```
1: procedure ScaleFrame(originalFrame)
2:     for pixel with index i in originalFrame do
3:         Create new List
4:         if pixel has valid depth value then
5:             Add pixel to List
6:         end if
7:         if pixel to the right for pixel has valid depth value then
8:             Add rightPixel to List
9:         end if
10:        if pixel to the right and down for pixel has valid depth value then
11:            Add rightDownPixel to List
12:        end if
13:        if pixel straight down for pixel has valid depth value then
14:            Add downPixel to List
15:        end if
16:        scaledFrame[i] ⇐ average depth of pixels in List
17:    end for
18:    Return scaledFrame
19: end procedure
```

8.3 Background Subtraction: Temporal Median Image

A model of the background is constructed during an initial phase of the sensing session, before any people enter the sensing area. The approach is called Temporal Median Image Subtraction and is also mentioned in the section "Background Subtraction in Depth Data". Temporal Median Image Subtraction was chosen because it is a simple and well-performing background subtraction approach that fits the contexts this body tracker is used in. The algorithm for constructing the temporal median image is shown in Algorithm 2. Initially, the first frames captured by the Kinect sensor are consecutively stored. Experiments showed that including the first 30 frames was sufficient. After the 30th frame has been stored, the corresponding pixels in the stored frames are collected in the same array and sorted. The shallowest pixel depth in the array is selected to be used in the background model frame. Finally, the noise in the background model frame is smoothed with a median filter.

Algorithm 2 Calculating the temporal median background model

```
1: procedure CalculateTemporalMedianImage()
2:     Create new List
3:     for each pixel with index i in a frame do
4:         for each frame in storedBackgroundFrames do
5:             pixelDepth ⇐ depth of pixel at i in frame
6:             List ⇐ add pixelDepth
7:         end for
8:         Sort List
9:         shallowestDepth ⇐ shallowest depth from sorted list
10:        temporalMedianImage[i] = shallowestDepth
11:    end for
12:    filteredTemporalMedianImage ⇐ process temporalMedianImage with median filter
13:    Return filteredTemporalMedianImage
14: end procedure
```

After the initial phase, the background model frame is subtracted from all new arriving frames. The subtraction is a computationally very light process. Each pixel in the arriving frame does only need to be looked up once and compared to its respective pixel in the background model. If the pixel in the new frame is lower than the respective pixel in the background model, the pixel depth is set to the maximum sensing depth (classified as background pixel).

If an object in the scene, that is not a person, is permanently moved during the sensing session, the earlier constructed background model becomes inaccurate. Fortunately, so far during test sessions this hasn't disrupted the later detection of people in the frame.

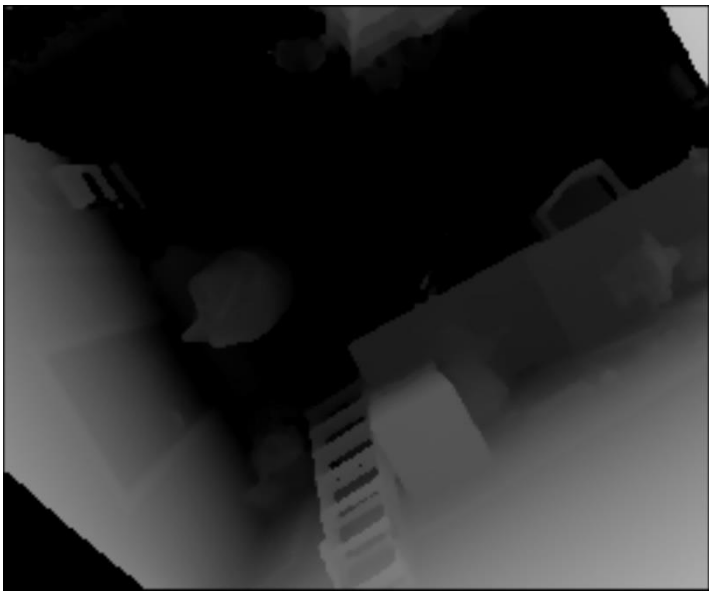


Figure 8.3 Temporal Background Model Image of a scene.

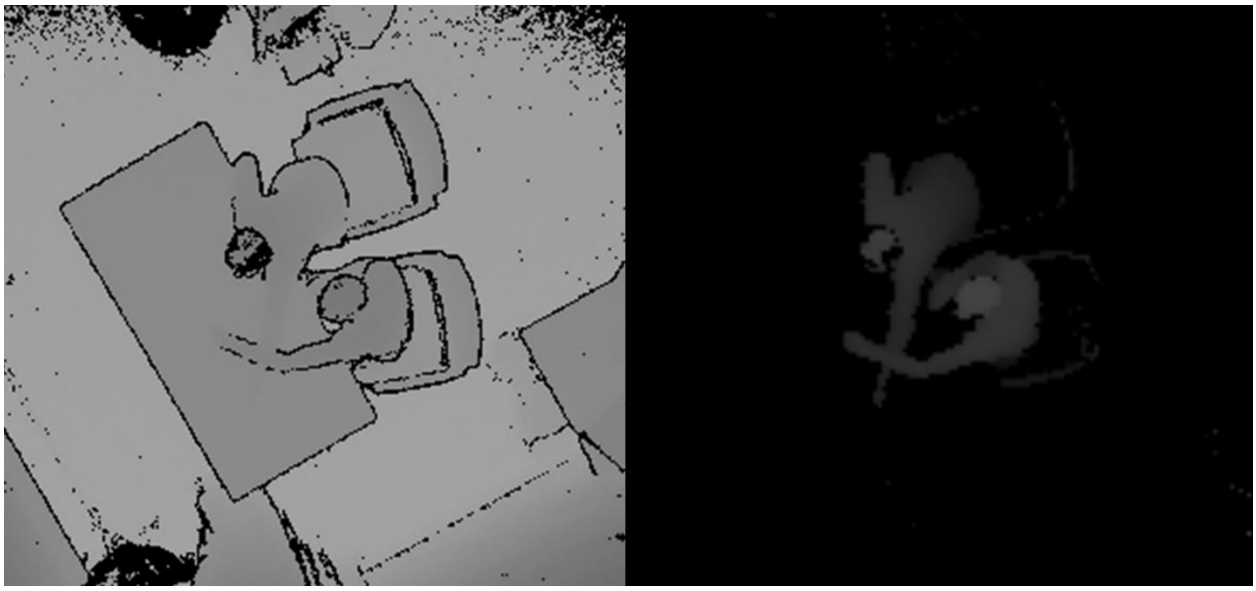


Figure 8.4 Background Subtraction. Left: the original depth frame. Right: The scene with the background subtracted.

8.4 Noise Removal: Median Filter

A depth frame is prone to have missing data points and the missing data points should be smoothed away before further processing. Median filtering is a frequently used filtering method in classic computer vision with RGB data, and its principle is easily adopted to depth data. A square kernel, also called a convolution matrix, is slid over each pixel of the input frame. The pixels in the kernel are sorted and as the name suggests, the median value in the kernel is usually selected. In my implementation, the best filtering performance was shown when an even smaller element, the element at index 2 (the indexing starts at 0), in the sorted kernel was chosen. The reason for this is that the missing data points are also included in the kernel, with depth values set to maximum sensing depth. Which means that by picking an index earlier in the sorted list (sorted ascending), the chances of picking a valid depth value are increased. However, if an index too early in the list is chosen, the output image will contain unnecessary amounts of spikes. The kernel can have different sizes, and for my implementation a kernel size of 3x3 showed itself to be the best tradeoff between computational load and filtering performance. The median filter algorithm is shown in Algorithm 2.

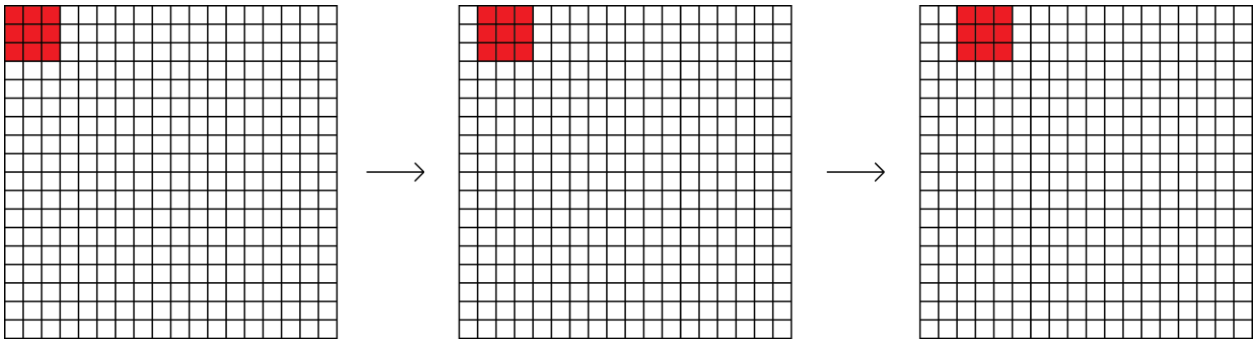


Figure 8.5 The square kernel is slid over the frame.

Algorithm 3 Median filter with a 3x3 kernel

```
1:  procedure MedianFilter3X3(originalFrame, elementIndex)
2:      kernel ⇐ create new List
3:      for each pixel in originalFrame with index i do
4:          kernel ⇐ add pixel
5:          neighbors ⇐ get eight surrounding pixels of pixel
6:          kernel ⇐ add neighbors
7:          if kernel contains valid depth data then
8:              Sort kernel
9:          end if
10:         selectedPixel ⇐ the pixel in the sorted kernel at the provided elementIndex
11:         filteredFrame[i] = selectedPixel
12:     end for
13:     Return filteredFrame
14: end procedure
```

Figure 8.6 Index in sorted kernel. The filtered results from choosing different indexes in the sorted kernel. Left: Index 2. Middle: Index 4. Right: Index 8. The person in the frame has shiny black hair and we see that if a high index is chosen depth data will be lost.

Median filtering an image is a computationally heavy task. For each pixel in the frame, the eight surrounding neighbor pixels and the pixel itself need to be added to an array and sorted. The average case performance when using quicksort as a sorting algorithm is

$$O(m * (n * log(n))) \tag{3}$$

where *m* is the number of pixels in the image and *n* is the total kernel size. As can be seen from equation 3, it is the sorting that creates the computational load. To increase performance, an optimization feature was created. Because the background is already subtracted from the input frame, the frame will contain substantial areas of background pixels without useful depth information. To reduce computational load, a helper method was created to check if the kernel contains valid pixel. If the kernel does not contain valid pixels, it moves on to the next pixel without sorting. The performance gain from the optimization was measured with the Stopwatch class which is a class included in the .NET framework that accurately measures elapsed time. The performance gain is showed in Table 3. Also, other faster implementations of the median filter was considered (Huang, Yang, & Tang, 1979)(Perreault & Hébert, 2007), but adaptation to depth values was not straight forward and wasn't prioritized in this thesis.

	With optimization	**Without optimization**
MedianFilter (milliseconds)	9.6	15.11
Total application (milliseconds)	44.85	47.07
Amount of total computational time	21.40%	32.10%

Table 3: The computational time of the median filter with and without optimization. The time is the average for all frames in a 5 min sensing session.

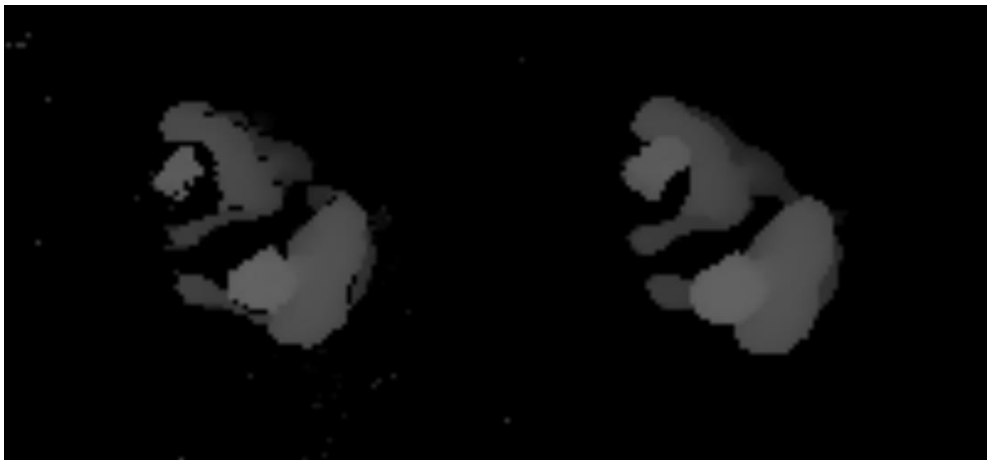


Figure 8.7 The foreground area in the frame before (left) and after (right) median filtering.

8.5 Head Detection

8.5.1 Haar-like Features

The concept of Haar-like features was first presented by Viola & Jones (2001) and has since been extensively used in several object recognition tasks, especially face recognition. The first real-time face detector was based on Haar-like features(Viola & Jones, 2001). A Haar-like feature is a detection window that consists of subsections of adjacent rectangular regions. The average pixel intensity in each region is calculated and compared across regions. The characteristics of the differences in pixel intensities is used in classification of the areas in the input image. The composition of subsections in the Haar-like features can vary, depending on the object feature that is to be detected.

Even though Haar-like features traditionally have been used for detection in RGB-images, the principle can easily be adapted to depth data. Inspired by the "center-surround" Haar-like feature presented by Lienhart et al. (2002), a Haar-like feature was developed for depth data that takes advantage of the characteristic strong depth gradients that usually exists around a person's head. As shown in Figure 8.8, the difference between the average depth in the outer rectangle and the inner rectangle is compared to a threshold. If the difference is above the threshold, the region inside the inner rectangle is considered to be a candidate head region and passed on to further validation. Also other Haar-like features was tried, for instance features with a round inner detection window. Yet, it was the detection window shown in Figure 8.8 that gave the best overall performance. The algorithm for the sliding detection window with the Haar-like feature is shown in Algorithm 4.

In other object recognition tasks, several passes are performed with different sizes of the detection window. For the current sensing setup, the heads do not vary much in size so only one pass with a fixed calculated size of the detection window is necessary. The size of the inner detection window is determined as the projected size into camera pixels of the average head diameter of 17.5 cm at a height of 170 cm (Algazi, Avendano, & Duda, 2001).

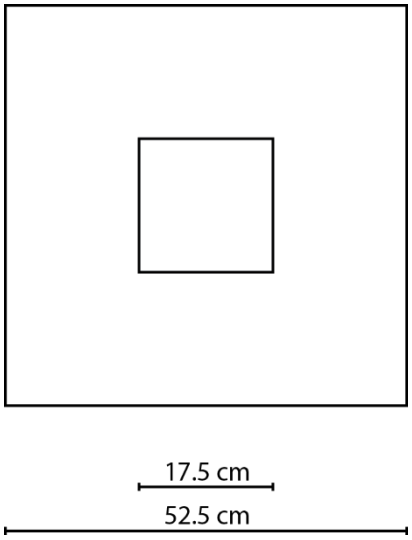


Figure 8.8 The Haar-like feature used to detect heads in the frame.

Figure 8.9 Haar-like window detection. The detection windows exceeding the threshold as they pass over a head in the frame.

Algorithm 4 Sliding detection window with Haar-like feature

```
1: procedure SLIDEWINDOW(frame, integralImage, differenceThreshold)
2:     candidatesList ⇐ create new List
3:     for each pixel in frame do
4:         Upper left corner of the inner detection window → coordinates of pixel
5:         innerWindowCoord ⇐ coordinates for inner detection window
6:         outerWindowCoord ⇐ coordinates for outer detection window
7:
8:         sumInnerWindow ⇐ get sum from integralImage and innerWindowCoord
9:         sumOuterWindow ⇐ get sum from integralImage and outerWindowCoord
10:
11:        averageDepthInner ⇐ Calculate average depth in inner window
12:        averageDepthOuter ⇐ Calculate average depth in outer window
13:
14:        if ((averageDepthOuter − averageDepthInner) > differenceThreshold) then
15:            headCandidate ⇐ highest point in inner window
16:            candidatesList ⇐ add headCandidate to List
17:        end if
18:    end for
19:    Return candidatesList
20: end procedure
```

8.5.2 Integral Image

The Haar-like features becomes a powerful approach when they are used in combination with integral images, also called summed area tables. Integral images are two dimensional lookup matrices that enables the area of a rectangular area in an image to be calculated very quickly and efficiently. The integral image has the same size as the input frame and each element in the integral image contains the sum of the pixels located on the up-left region of the input frame. Thereby reducing the procedure of computing the sum of an area to only four lookups in the integral image:

$$Sum = C - B - D + A \tag{4}$$

The integral image only needs to be calculated once for each frame, before the detection window with the Haar-like feature is passed over the frame.

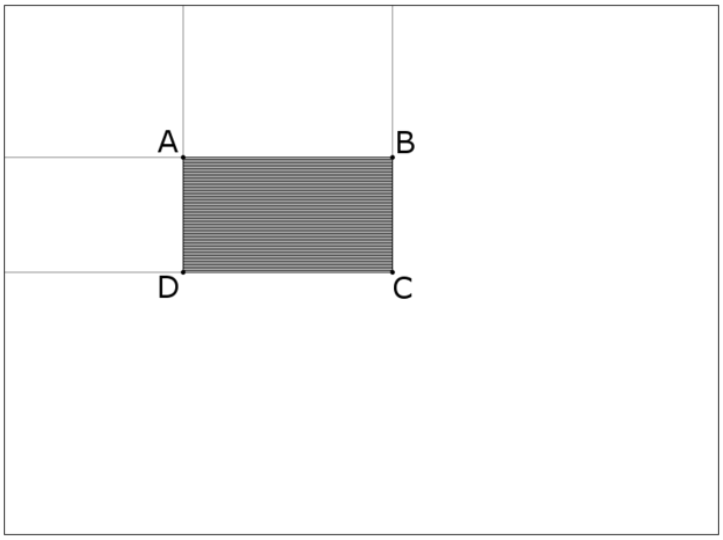


Figure 8.10 Integral image. The sum of the pixels in the gray area can be calculated quickly by C – B – D + A in the integral table.

8.6 Classification

As the detection window is passed over the frame and the candidate head regions are located, the highest point in the regions are used to represent the regions in further processing. The highest point is chosen because it has a higher probability of being closer to an actual head, considering there are rarely cases where there is an object in the scene that is that close to the actual head, but still not a head. Further, if the center point of the inner detection window or a random point was chosen, there would be a risk that a pixel without valid depth information is chosen.

The next step in the detection pipeline is grouping the candidate head points together. All candidates that are within a certain Euclidean distance from each other in the XY-plane are grouped together. The grouping is implemented using a disjoint-set data structure, and it is the pixel with the shallowest depth (highest point) that is used as the set representative. After the grouping, only the representatives from the groups that are bigger than a certain threshold are kept.

At this stage in the pipeline, it is a good chance that the candidate points are very close to the actual heads in the frame. Still, experiments show that when the head region is challenging, such as

- *shiny black hair*

- *the region is located at the edges of the sensing area where the accuracy of the sensing is worse*
- *when several heads are located very close to each other*
- *when the person in the frame is leaning forward so the upper back almost is located at the same height as the head*

the candidate points may be inaccurate. To improve the accuracy, the advantages of the depth information is used. Each of the candidate points are passed to a function that finds the local maximum height by recursively searching through the neighborhood pixels for depths that is higher up than the candidate point. To avoid being stuck in a very small local maximum that could occur due to noise, the function gets the 5x5 neighborhood area in each recursive call.

8.7 Validation of Candidates

After all the highest points connected to the candidate head points are determined, the possible heads resulting from those highest points needs to be validated. As described in section "*Detection and Tracking in Top-View"*, several other approaches have been used to validate a 3D head shape candidate. In the current implementation several of those validators, and combinations of several, were tried. Creating a hemi ellipsoidal model like Tseng et al. (2014) and measuring the error between the head candidate and model, showed itself to be inaccurate when the head shape became inaccurate, which could happen when a person in the frame is tilting its head or when having a special hairstyle. The same was experienced with the template matching approach of Oosterhout et al. (2011). Evaluating movement in the candidate head regions across frames builds on the assumption that the head always has a certain movement. The approach became inaccurate when persons in the frame kept their head still, and created false positives when other objects were moved.

The validation approach that gave the best performance was simply evaluating the size of the head blob. The candidate head blob was found by implementing an iterative connected-component algorithm that uses a breadth first search to search all pixels connected to the starting point (Dillencourt, Samet, & Tamminen, 1992). The starting point is the highest point located in the previous stage, and to decide if the neighbor pixel should be included in the head blob or not, the height difference between the pixels are compared to a threshold. The threshold is set to 5 cm and the search depth is limited to 110 pixels to make sure the search stops at the edges of the head. Both an iterative and recursive implementation of the connected-component algorithm was tried, and the iterative had the best performance.

Figure 8.11 Head blob. A head blob after growing region with the connected-component algorithm and validation.

8.8 Geodesic Distance Map

After the heads of the people in the scene are detected, we also want to track more of the bodies. To successfully include further functionality of precise heat maps and energy calculations, tracking of the whole body is necessary.

Tracking a human body in natural environments is a very complex problem. A human body has about 244 degrees of freedom and can adopt a range of different poses (Vladimir Zatsiorsky & Boris Prilutsky, n.d.). A great variety of approaches of tracking in depth data has been presented in literature since the arrival of the first Kinect. The most successful approaches rely on using machine learning techniques on large datasets of human poses (Jamie Shotton et al., 2013). Still, training a machine learning algorithm for human poses, can be very resource demanding.

From a top-view perspective the amount of poses a human can adopt are more limited than from a side, back or front view. The challenge of tracking a body-pose basically reduces itself to track hand-motions and leaning-motions. To achieve this, a geodesic distance graph is created on the surface of the body. An approach that was also used in Plagemann et al. (2010) and Tseng et al. (2014).

A geodesic distance graph is created by using the depth data to calculate the Euclidean distance in 3D space between neighboring pixels. If the distance between the pixels are small enough, they are connected in the geodesic graph. The graph becomes a mesh of connected points on the surfaces of the foreground objects in the scene. We only want the geodesic graph of the bodies in the scene, so in case the background subtraction cannot completely remove all

background pixels, an optimization is implemented where only points closer than 1 meter in Euclidean distance from the closest validated head point are considered.

Algorithm 5 Create geodesic graph on the surface of the foreground objects

```
1: procedure CreateGeodesicGraph(foregroundFrame)
2:     for each pixel in foregroundFrame do
3:         if pixel has valid depth value then
4:             neighborList ⇐ get eight surrounding pixels of pixel
5:             for each neighbor in neighborList do
6:                 if neighbor has valid depth value then
7:                     if distance between neighbor and pixel is lower than threshold AND distance
   to closest head candidate is lower than threshold then
8:                         Add neighbor as a connecting node for pixel
9:                     end if
10:                end if
11:            end for
12:            if neighbor has connecting nodes then
13:                Add neighbor to geodesicGraph
14:            end if
15:        end if
16:    end for
17:    Return geodesicGraph
18: end procedure
```

Figure 8.12 The geodesic graph on the surface of a person.

Because the geodesic distances along the surface mesh between the body regions are nearly constant independent of pose, the body regions can be identified by finding the shortest paths from the highest point on the validated head to the other parts of the body. The shortest path search is performed using the classic well-performing Dijkstra shortest-path search algorithm.

Figure 8.13 Body regions. The body regions are found by finding the shortest path from the head to the rest of the body. The distance to the body regions were found heuristically.

8.9 Hand detection

The hand locations can also be estimated once the geodesic distance graph is established. The detection approach for locating hands show reasonable performance when the persons in the scene are located by a table, and a depth threshold can be set at the height of the table. If there is no depth threshold to exclude the lower parts of the body from the waist down, the hand regions are confused with the areas just below the waist that has the same geodesic distance to the top of the head as the hands (see Figure 8.14).

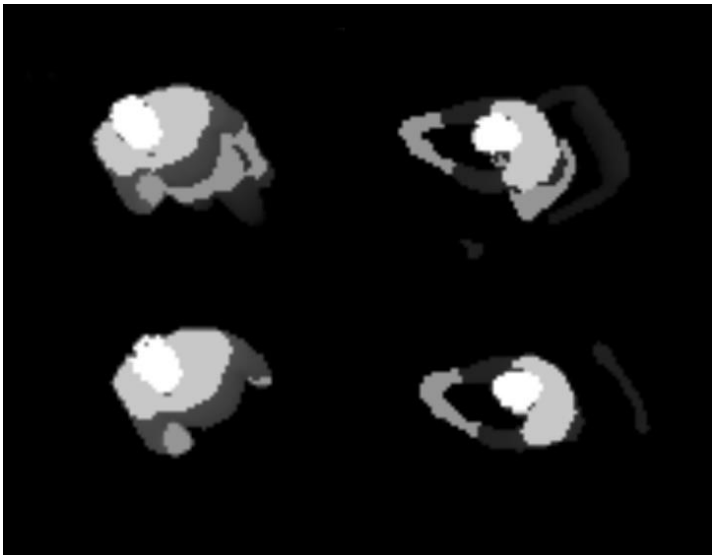


Figure 8.14 Hand detection with and without depth threshold. The defined "hand-region" in dark gray. Without depth threshold (up) and with depth threshold (down.)

The hand detection could be improved by including the RGB data from the Kinect sensor and using hand color in the region validation. Another approach would be separating the hand candidate regions extracted from the geodesic graph, then performing a region growing algorithm on the candidates and finally performing a shape analysis of the grown regions as validation. Yet, in the thesis, other features needed to be prioritized over hand detection.

8.10 Tracking

The location of the bodies are detected in each frame, therefore no dedicated tracking algorithm is used. When bodies are detected in a new frame, their head locations are compared to the average head locations over the last five frames of previously identified bodies. The average values are used to get a more stable tracking of the bodies. Experiments showed that if only the location from the last frame was used, the tracking would be erratic and imprecise.

The new bodies are given the Id of the previous body that is closest to their location, as long as the previous body isn't closer to any other new body. The algorithm is shown in Algorithm 6. This approach assumes that the bodies stay in the frame for the period of the sensing for correct identification of new bodies. Still, the case of bodies leaving the frame could be included with simple modifications.

Algorithm 6 Match the detected new candidate heads with bodies from previous frames

```
1: procedure IDENTIFYHEADS(candidateHeads)
2:     recentBodyIds ⇐ get the body ids from the 5 most recent frames
3:     avgLocationsRecentHeads ⇐ get the average locations of the heads with recentBodyIds
4:     distancesFromCandidateHeadsToRecentHeads ⇐ calculate the distances from all
       candidateHeads to all recent heads
5:     sortedDistances ⇐ sort distancesFromCandidateHeadsToRecentHeads by ascending dis-
       tance
6:     identifiedHeads ⇐ Create new List
7:     for each distance in sortedDistances do
8:         if candidateHead and the id belonging to a distance is not in identifiedHeads then
9:             identifiedHeads ⇐ add candidateHead with id
10:        end if
11:    end for
12:    for each candidateHead in candidateHeads do
13:        if identifiedHeads does not contain candidateHead then
14:            identifiedHeads ⇐ add candidateHead with id=-1
15:        end if
16:    end for
17: end procedure
```

8.11 Mechanical Energy Estimation

Estimating mechanical energy could not be done with traditional RGB cameras. With depth cameras, the movement in the 3D space of the tracked person can be directly measured. According to previous research, the most promising approach that would be possible with a Kinect is considered to be calculating the energy change in each body segment separately before adding each to a total (Van de Walle et al., 2012). This is also the approach that is used in the thesis.

The same assumptions is made about the mass distribution for the segments in the human body as Liu et al. (2012) and are shown in Table 4. The weight of the person is either set to an average value of 65 kg or defined before the tracking session to match the body weight of the person that will be tracked. The change in mechanical energy is then estimated real-time between consecutive frames. The displacements $disp$ of the average center points of the head, torso and hands are calculated for each XYZ-direction from one frame to the next. Timestamps are collected from the Kinect API to calculate the Δt between the frames. For each body segment (seg), the total mechanical energy change is then calculated by the equations for kinetic energy (5,6) and potential energy (7) and summed together.

$$E_{TotalX} = \frac{1}{2} m_{seg} \left(\frac{disp_x}{\Delta t} \right)^2 \tag{5}$$

$$E_{TotalY} = \frac{1}{2} m_{seg} \left(\frac{disp_y}{\Delta t} \right)^2 \tag{6}$$

$$E_{TotalZ} = m_{seg} G * disp_z \tag{7}$$

$$E_{Total} = E_{TotalX} * E_{TotalY} * E_{TotalZ} \tag{8}$$

At the end of the tracking session, the already stored energy changes during the session (Joule/frame) are filtered through a one-dimensional median filter to remove spikes from tracking instabilities before the energy is plotted against time. Also, the average change in

mechanical energy per second (Joule/s) throughout the whole session is calculated and displayed (Figure 8.15).

Body Segment	Head	Forearm	Upper arm	Thigh	Shank	Torso
Amount of Total	**23.1%**	**1.8%**	**3.5%**	**9.4%**	**4.2%**	**58%**

Table 4: Assumed distribution of mass in the body.

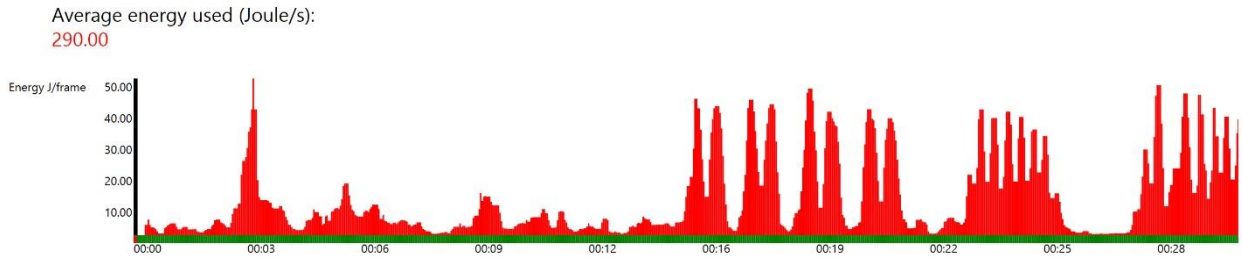


Figure 8.15 The display of the energy-use after a session. The X-axis is green when the tracking of a person was successful in the respective frame.

Some challenges with the approach has been experienced and should be mentioned. Because the placement of the sensor is top-view, very little data about the lower part of the body is visible to the sensor and contributes to the estimates for the energy expenditure to be incomplete. Also, the average center points used for each body segment in the energy calculations will tend to be a bit closer to the sensor than the actual center of mass. The reason is that the center points are calculated from the pixels of the body segment visible to the sensor, which are the pixels on the upper side of the body segments. Moreover, the transfer of energy between adjacent body segments are not taken into account, resulting in an estimate higher than the real value (Van de Walle et al., 2012). Finally, an important aspect is that the negative energy changes, for instance when the potential energy of a body segment is lowered, are also included in the calculations. If compared to methods trying to estimate only positive energy exerted by the bodies, the results will differ.

8.12 Heat Map

Heat maps are graphical representations of data from a 2D matrix. The areas in the matrix where the interesting variable occurs are normally visualized with a color resembling varying

intensities of heat depending on the value of the variable. Usually with red as the color representing the areas with most heat. Typical uses of heat maps are eye-tracking, mouse-tracking on webpages, world maps with different variables and many more. With heat maps, data can be visualized in a very intuitive and clear way.

Since a geodesic graph is created on the whole surface of each person, an accurate heat map of the movements of the person can be created. For each new frame, the results from the detection are saved and the global heat map is updated. Each body id is assigned a unique color, and at the end of the tracking session the accumulated values in the heat map are normalized so the variation in heat is visualized by variation in intensity of the color.

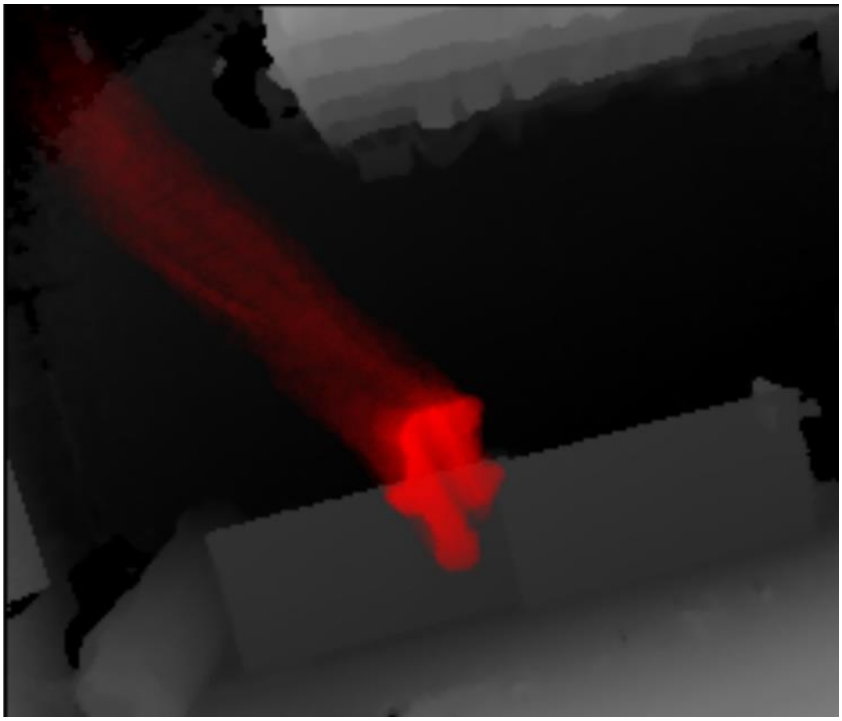


Figure 8.16 Heat map of a person that just entered the scene.

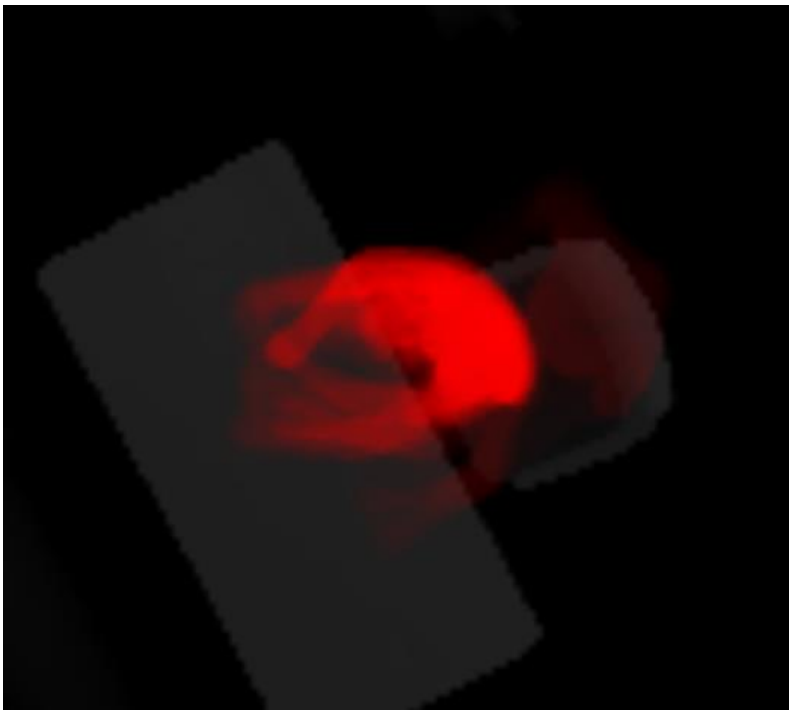


Figure 8.17 Heat map of a person sitting.

9 Validation and Tracking Success

9.1 Assumptions

These are the assumptions made in the tracking approach:

1. ***Sensor has a clear view of the people's heads:*** No objects should disrupt the sensors view of the people in the scene.
2. ***The background is relatively static:*** For the chosen background subtraction algorithm, Temporal Median Background, to work efficiently the background needs to be relatively static. If changes in the background occur during the sensing session, the tracking of people should not be affected, but the foreground will be more complex and may lead to marginally slower processing of the frames. Experiment show that the slower processing isn't noticeable to the human eye.
3. ***Average human head size:*** In the Haar-like feature that is used to detect the initial location of the head region, the size of its detection window is adapted to the hardcoded average size of the head of a human, which is 17.5 cm diameter (Algazi et al., 2001). Experiments show that using the average head size works well for several different people. Regardless of the imprecisions caused by assumptions in head size, for the initial head detection many other factors such as incomplete heads due to missing data points and variations in posture are also sources of imprecision.
4. ***Mass distribution of body segments:*** For the energy calculations, the application assumes the body to have an ideal distribution of mass.

9.2 Accuracy and Precision

9.2.1 Accuracy

The accuracy of the raw data arriving from the Kinect v2 has been evaluated in several research papers. Lachat et al. (2015) evaluated its usefulness for close range 3D modelling. Error sources such as pre-heating time, outdoor efficiency and influence of materials and colors were investigated, as well as deviations between measured and true distances. The conclusion was that the sensor measurements deviated ±5 mm in the recommended sensing range of 1.5-4.5 m, and that the achieved results looked promising. Butkiewicz (2014) measured the standard deviation of the depth measurements to increase linearly from 1 mm at 1.5 m to 3.5 mm at 4.5 m. Yang, Zhang, Dong, Alelaiwi, & Saddik (2015) evaluated accuracy distribution, depth

resolution, depth entropy, edge noise and structural noise in the Kinect v2. A cone model of the accuracy at certain distances was obtained (Figure 9.1). The green and yellow zones show good accuracy while the red zones have a depth accuracy error that exceeds 4 mm. In the mentioned papers, the imagined applications demand high accuracy from the sensor. For the application in the thesis, body tracking and estimating mechanical energy, the accuracy is considered to be satisfactory in the complete field of view of the sensor.

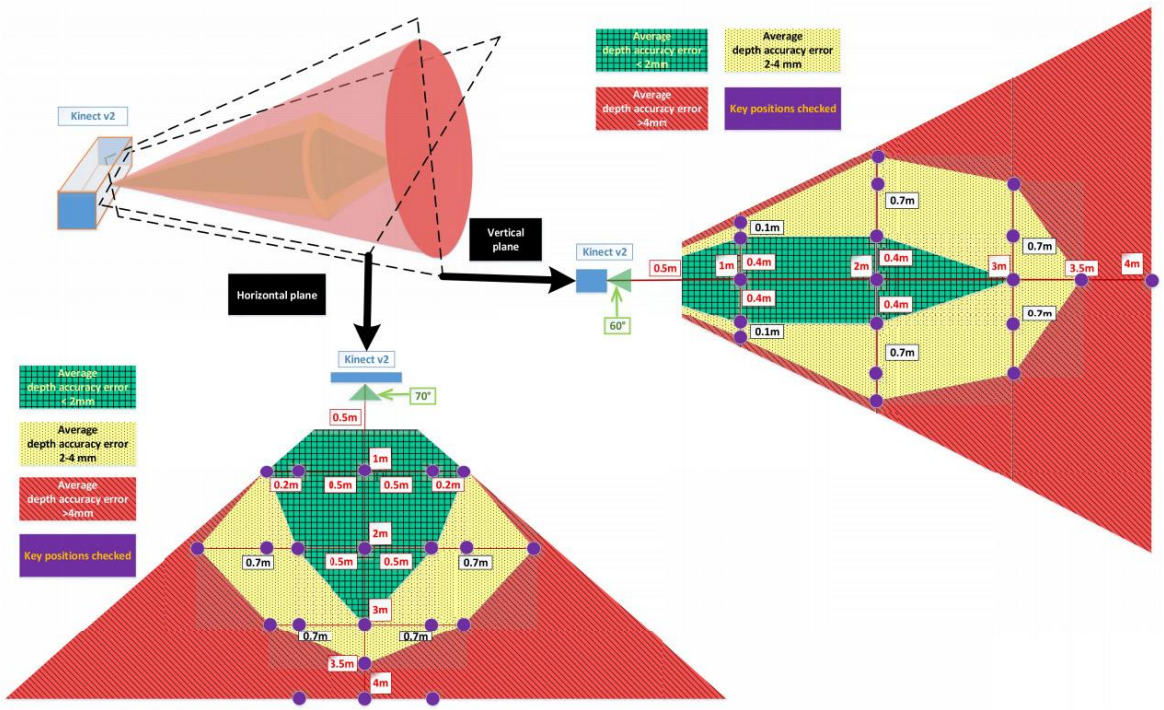


Figure 9.1 Accuracy throughout the field of view for Kinect v2. Source: Yang et. Al (2015)

9.2.2 Precision

To assess the precision of the tracking, a *Logger class* was developed that projects the results of the tracking in real-time on the monitoring window where also the point cloud of the depth data is projected onto. From visual inspection, the tracking of the head and torso show stable precision when tracking is present. Further, a *TrackingDiagnostics class* was implemented that calculates the total amount of the frames where bodies were detected successfully. In total, the application was tested on 5 different people, both male and female and weights ranging from 50 to 85 kg.

Figure 9.2 Tracking results of head and torso showed in monitoring window.

The tracking of the hands show more variable results. It is apparent that using geodesic distances alone is not sufficient to get stable and reliable hand tracking. The two main issues that occur is that the hand regions get confused with the region around the waist of the person and that areas on the body on the geodesic path connecting the hands to the head may become occluded. In these cases the geodesic distance to the hands change to be outside the defined distance or the hands don’t get included in the geodesic map at all.

9.3 Generally Challenging Scenarios

Some scenarios are more challenging in general for the tracking application and the Kinect than others. They may result in inaccuracy in the depth measurements, or most of the time missing depth data points. Approaches have been researched to meet these challenges and consist mainly of median filtering, morphological operations and more complex methods such as building noise models from the scene (V.-T. Nguyen, Vu, & Tran, 2015). Still, some issues are necessary to be aware of.

9.3.1 Materials and Color

Reflective and very light absorbing materials weaken the intensity of the infrared rays more than other materials, resulting in the sensor calculating a larger distance than reality or won't be able to give a depth measurement at all (Lachat et al., 2015). Examples of such materials are shiny black hair or objects covered with carbon black materials. This issue was especially visible in my thesis because most of my testing was performed with Chinese. No other hair color or style than vey shiny black hair seemed to be an issue.

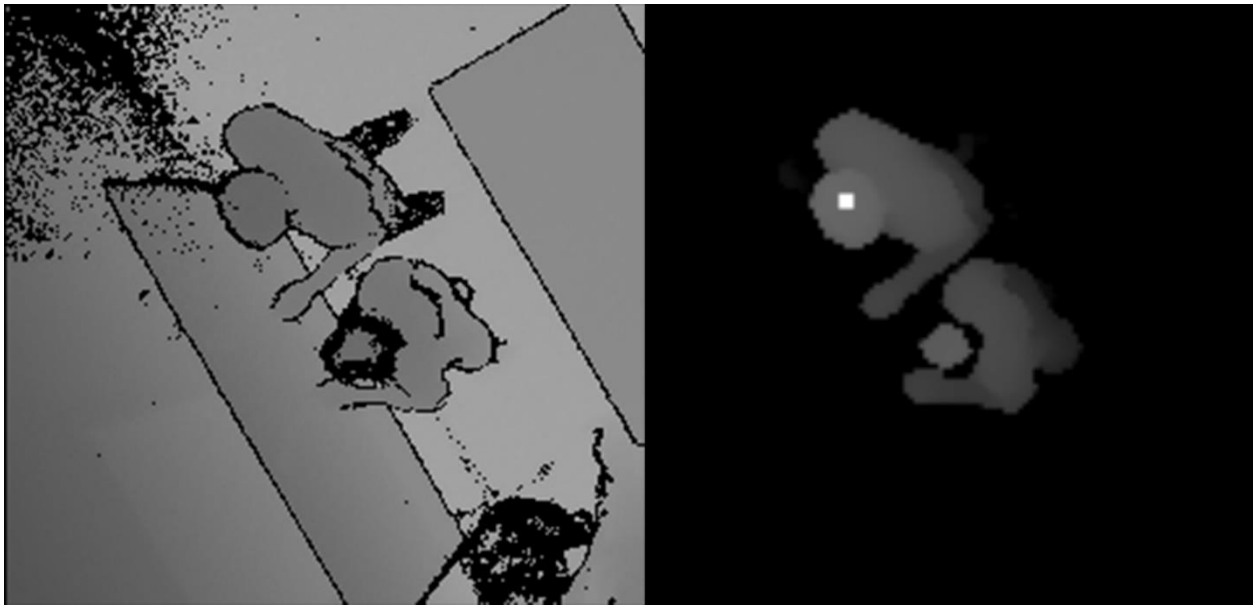


Figure 9.3 The tracker has problems detecting the girl with black hair. The raw depth image on the left shoes that a large portion of the head of the girl lacks data points.

9.3.2 Flying Pixels

Flying pixels is a known problem in depth sensors using time-of-flight depth measurement technology. Flying pixels occur when a single pixel is covering the depth data of an edge where

both the light reflected from the object closer to the sensor and the object further from the sensor contribute to the same depth measurement. The pixel then gets a depth measurement somewhere in between the two objects (Butkiewicz, 2014).

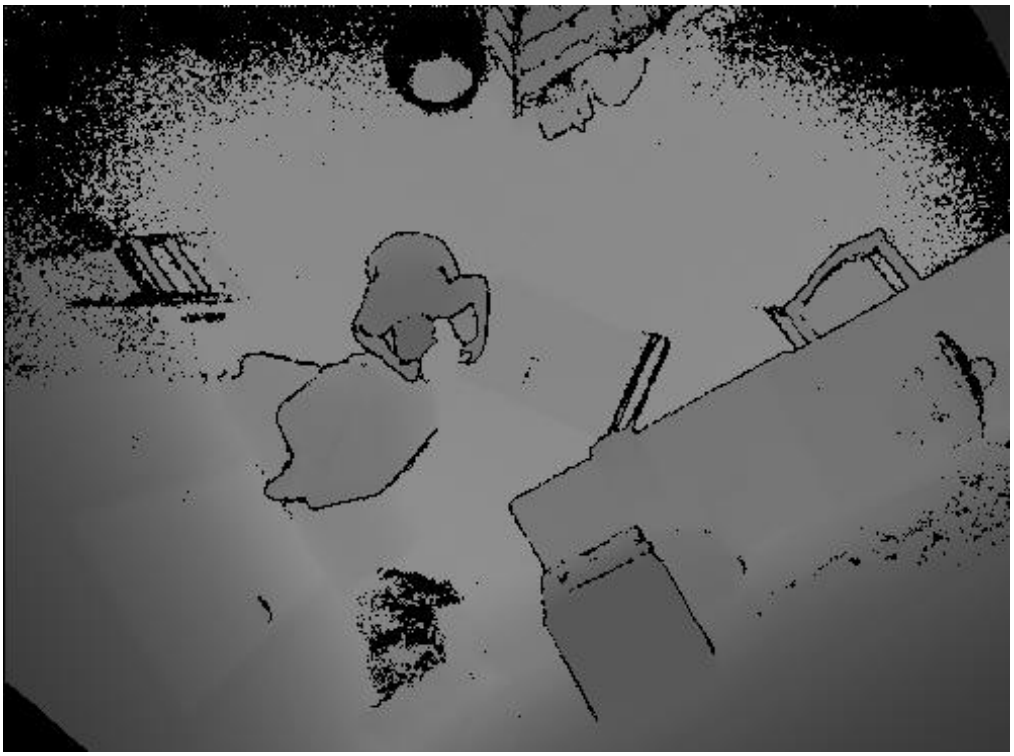


Figure 9.4 Flying pixels can be seen along the edges of objects.

9.3.3 Bodies Close to Each Other

When two or more bodies are located close to each other in the scene, most tracking algorithms meet challenges of classifying what features belong to what body. For the tracking approach developed in this thesis, several bodies close to each other may result in only one of the heads being recognized as a head, or if the bodies are in contact, the geodesic graphs may be confused between the bodies.

9.3.4 Less Depth Information in Outer Regions

The actual distance the infrared rays need to travel to give depth information in the outer regions of the frame are further than in the center. Thereby resulting in more missing depth data and inaccuracies in those regions than at equivalent depths in the center of the sensing area.

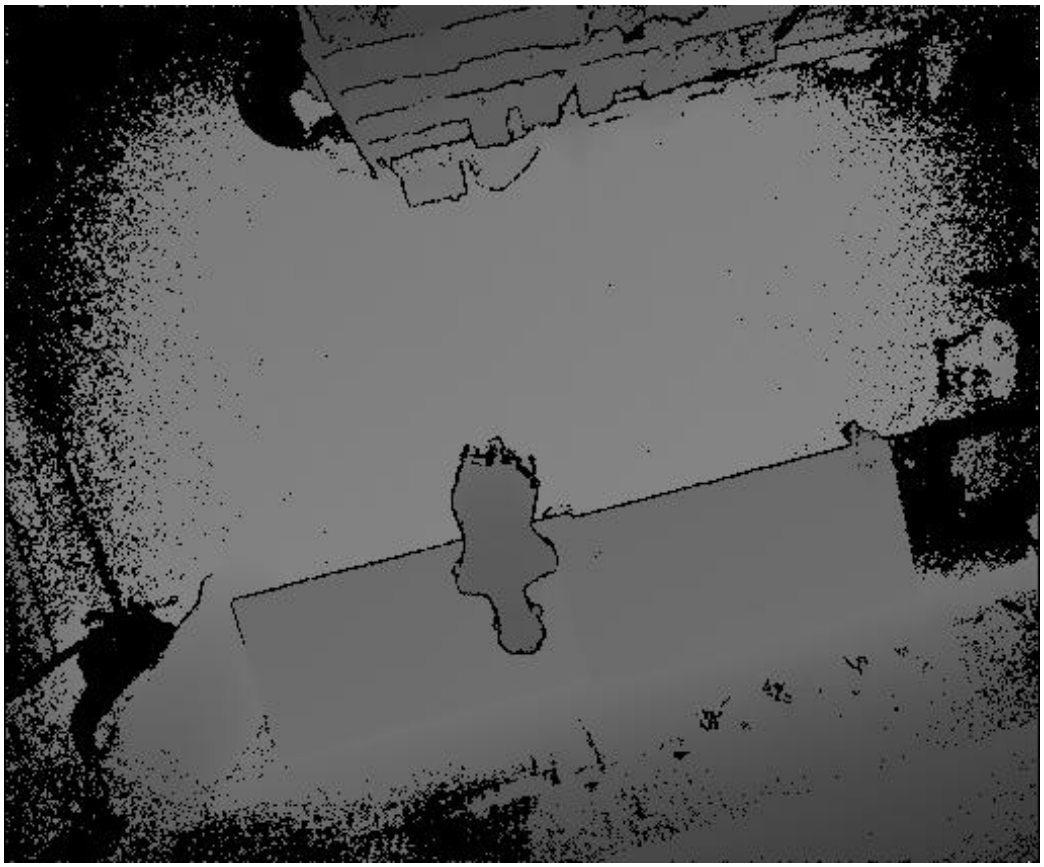


Figure 9.5 Outer regions of the scene contain less depth data.

9.4 Speed and Memory Performance

The tracking application needs to be able to process the frames fast enough to keep up with the data rate from the sensor and Kinect Studio. If the frames aren't processed at framerates close to the framerate arriving from the sensor, the application will lose information from the frames missed. The framerate arriving from the sensor is 30 FPS. The measured average processing framerate of the application is $\approx 26\ FPS$ without the monitoring window and $\approx 22\ FPS$ with. This framerate is perceived as real-time and sufficient.

The application was implemented in C# and all testing has been performed with the following computer specifications:

- 2.7 GHz dual-core Intel Core i5 processor (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache
- 8GB of 1866MHz LPDDR3 onboard memory
- Intel Iris Graphics 6100

The main bottlenecks for the processing speed are identified by using the performance and diagnostic tools (Figure 9.6) in Microsoft Visual Studio 2013:

- Searching through the geodesic map with the Dijkstra shortest path algorithm.
- Sliding the Haar-like detection window over the image.
- Median filtering the frame.

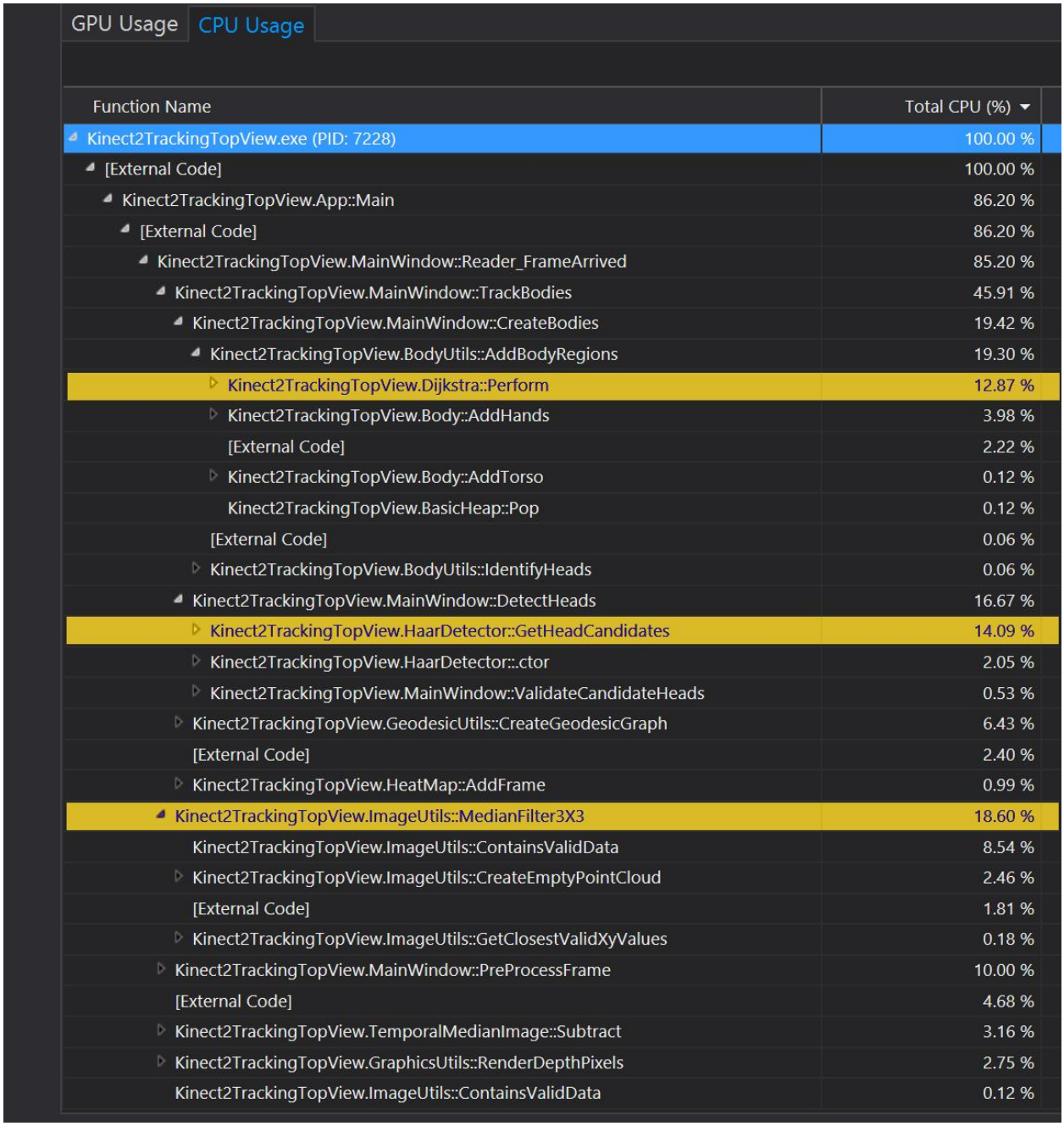


Figure 9.6 The CPU usage of the tracking application.

The memory use is also analyzed to make sure the application isn't leaking any memory and doesn't have a memory footprint that is too big. The average memory use is just below 120 MB as shown in Figure 9.7.

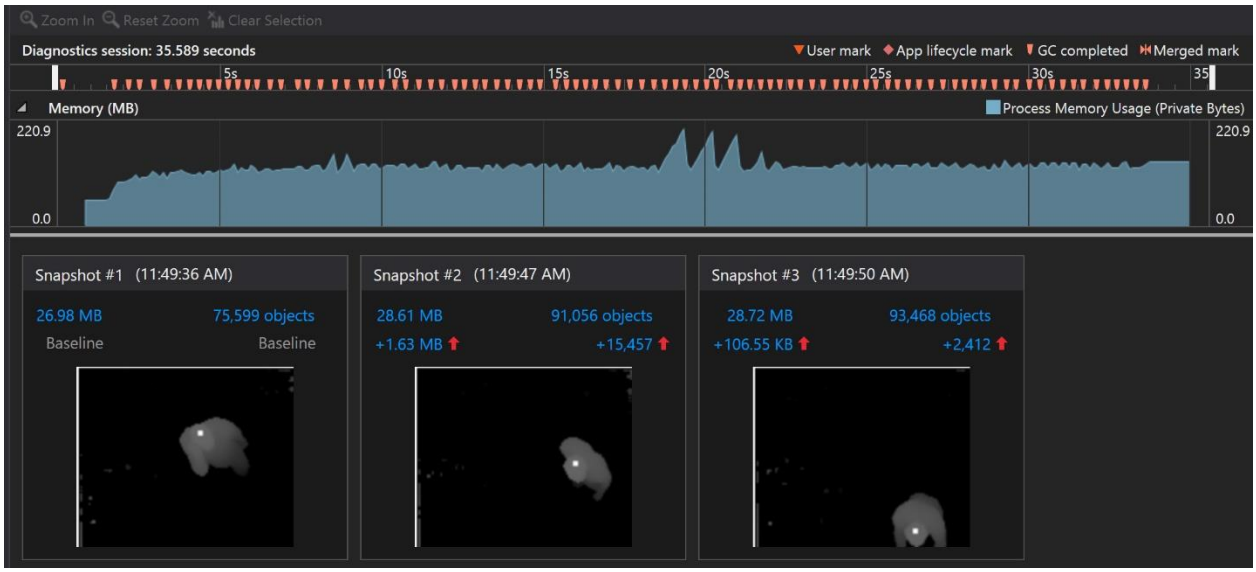


Figure 9.7 The memory use of the application.

A *Logger class* was developed early to be able to continuously and easily measure the processing time of added features in the application. Clean and readable code was emphasized through the whole project, although some sacrifices had to be made in favor of speed, such as using arrays with predefined size instead of dynamic lists when collecting neighbor pixels. The time in milliseconds for the most important features are shown in Table 5.

Feature	**Time(milliseconds)**
Down-scaling frame	≈ 5
Median filtering	≈ 10
Sliding detection window with Haar-like feature	≈ 9
Create geodesic graph	≈ 4

Add body regions	≈ 8
Other	≈ 7
Total	≈ **43**

Table 5: The time used by the feature in the application.

10 Tracking Application in Realistic Design Context

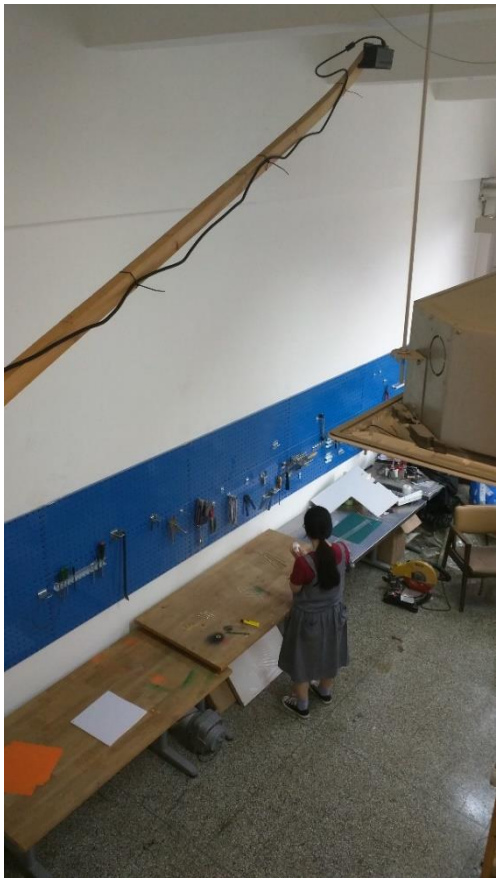
10.1 Egg-Drop-Challenge

To evaluate the developed sensing application in a realistic design context a prototype of an experiment was developed. The experiment was chosen to be a version of the well-known egg-drop-challenge. In the egg-drop-challenge the participants were asked to design a way, with the provided materials, to protect a raw chicken egg from breaking as it would be dropped from increasing heights onto a concrete floor. The egg-drop-challenge was chosen because it provides a realistic scenario of early prototyping in early-stage-design, as well as being simple, fun for the participants and doesn't take up too much of the participants' time. Because the experiment was conducted at the end of the semester, there were very few possible participants at the university location.

The participants performed the experiment one by one and were given the exact same instructions. All participants were aware that they were recorded and that the recording would be analyzed and used in this thesis. None of the participants were familiar with the egg-drop-challenge before the experiment.

Each of the participants was given 10 min to perform the challenge, and would be informed about remaining time every minute and when there was 30 seconds left. The materials that was provided were:

- 5 sheets of normal paper
- 1 30x30 cm poster board
- 1 75x15 cm Styrofoam sheet
- 8 rubber bands
- 6 Q-tip
- 1 m string
- 6 wooden sticks
- 30 cm tape
- Scissors, ruler, glue and a pencil



Four individuals participated in the experiment. Unfortunately, one of the participants had very long shiny black hair that created too many missing data points for the tracking to be precise enough. The results from the other three sessions are shown in Figure 48, 49, 50.

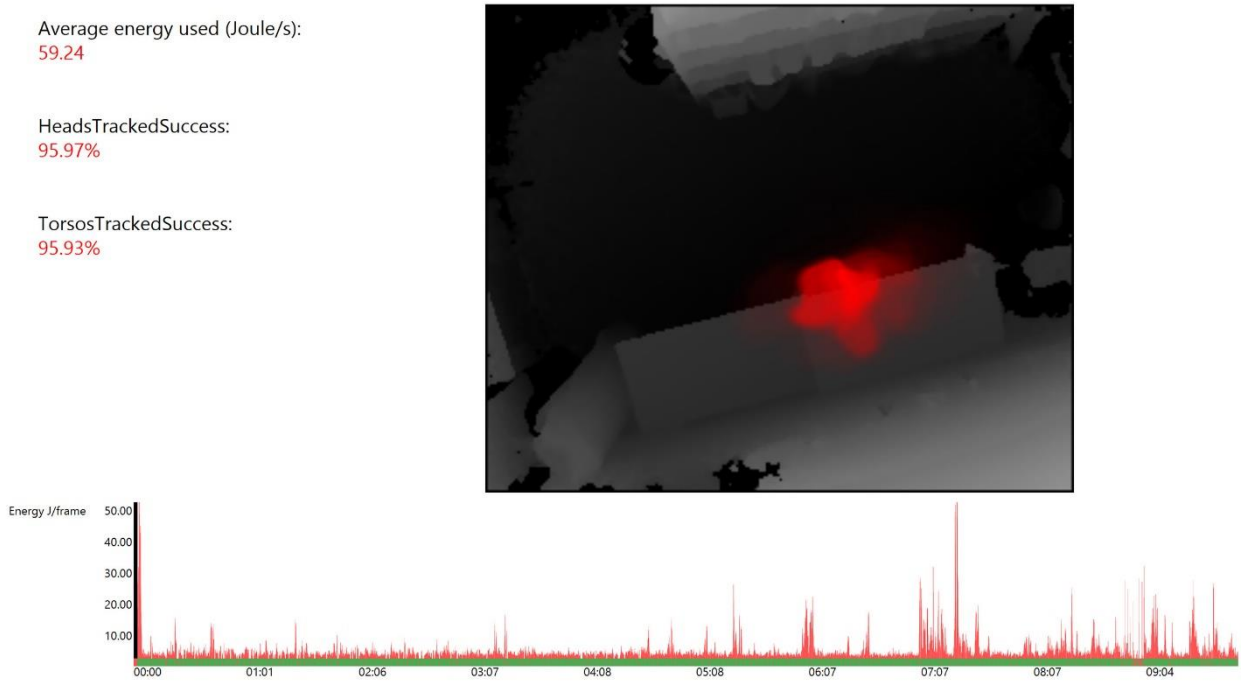


Figure 10.1 The results of participant one in the egg-drop-challenge.

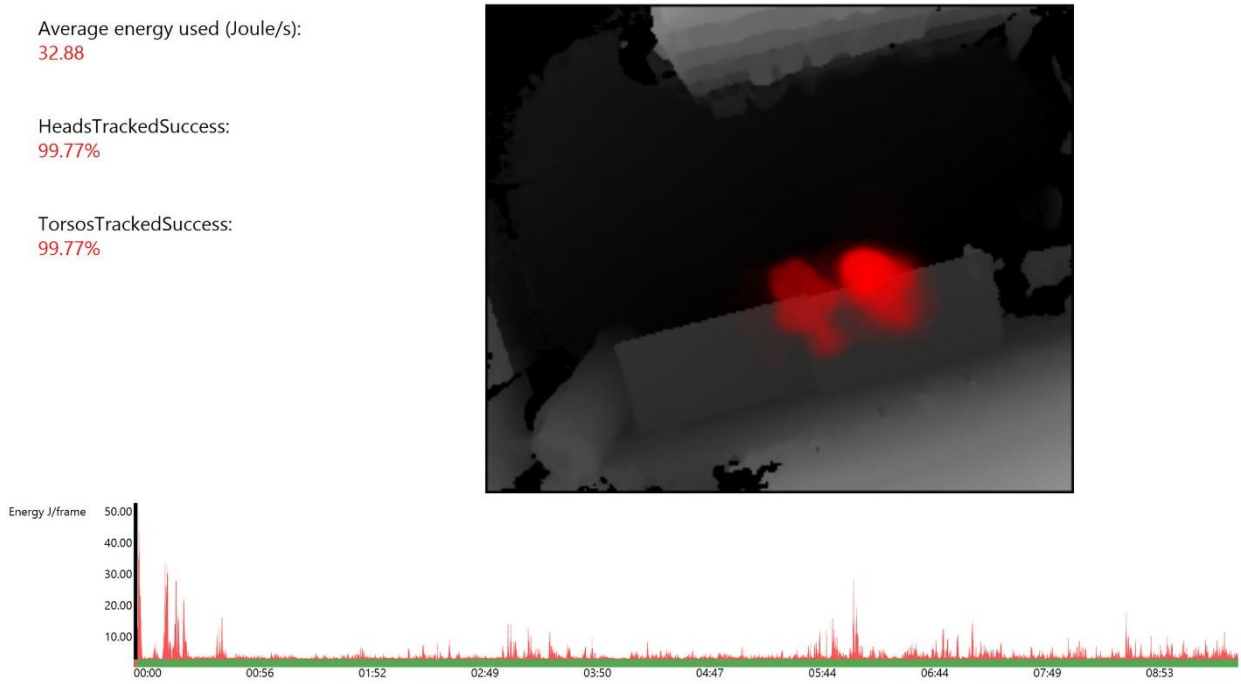


Figure 10.2 The results of participant two in the egg-drop-challenge.

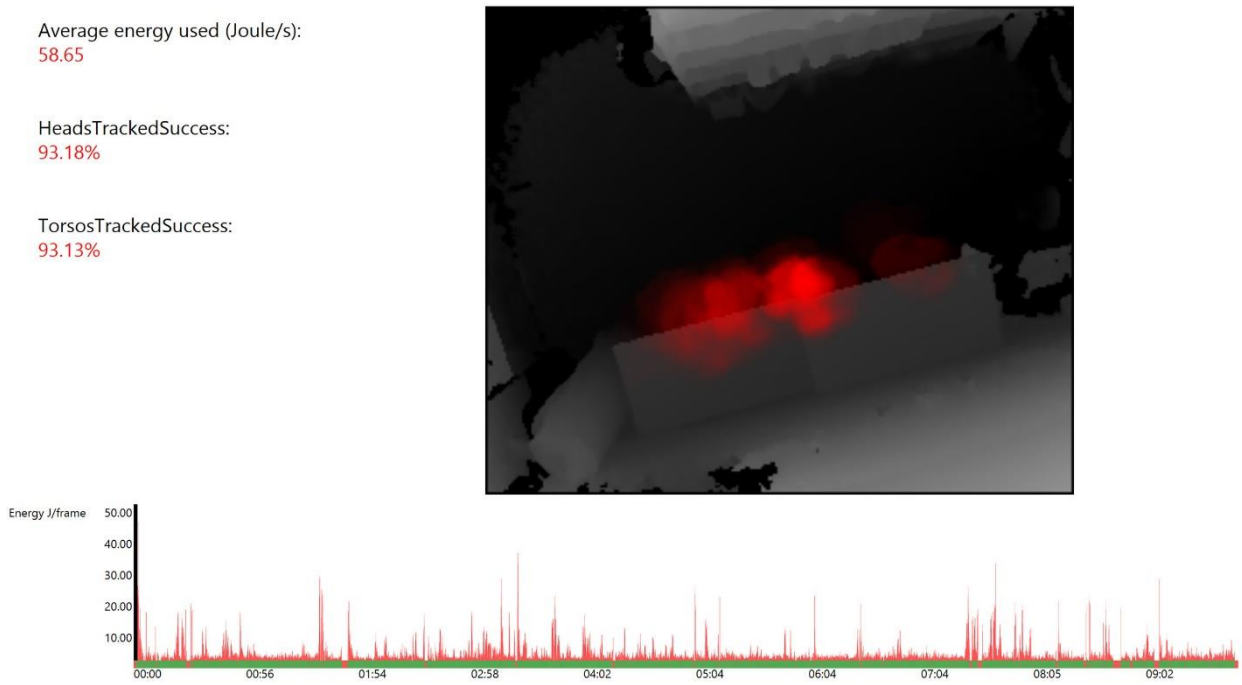


Figure 10.3 The results of participant three in the egg-drop-challenge.

Visual inspection while running the application showed that the tracking of the head and the torso worked well in all three sessions. Tracking of the hands were to imprecise and was excluded from the energy calculations. The main reason for the imprecisions was that the hands were occluded by the upper body and head most of the time (Figure 10.4). Even though participants one and three both had shiny black hair, similar to the excluded participant, the hair-styles were shorter and not as straight, resulting in the tracking showing stable behavior. Still, participant number two had brown hair and got a more complete depth cloud as well as more precise tracking, as can be seen by both the measurement of the tracking success and the green bar in the energy graph indicating if tracking is present or not.

Figure 10.4 Hand occlusion. From the top-view the hands were often occluded during the egg-drop-challenge.

10.2 Energy Calculations

Participant one did not move as much as participant three while working, however, because of much higher body weight of 81 kg compared to 55 kg the estimated average mechanical energy used was almost the same. Participant two was also very stationary, but has a body weight of 50 kg, resulting in a much lower estimate for mechanical energy use. The energy graphs show a very small constant base noise that probably influenced the average calculations a bit. The noise is a result of the small differences in the depth cloud of the persons because of sensor imprecisions, which results in the average head and torso points moving a bit between frames. The noise is still almost non-existent in the graph of participant two with brown hair and a more stable tracking

10.3 Heat Maps

The heat maps show the locations and positions each participant was working in most. Participant one had a relatively stationary working location indicated by one red silhouette. Participant two had two main working locations during the experiment, one was used more than the other as seen in the difference in color intensity. While participant three was the most mobile and working at several locations by the designated tables.

10.4 Reflections on Experiment

The experiment was tested beforehand by the author, who felt there was substantial time pressure with only 10 min. In the real experiment on the other hand, three of the participants implied they were finished before the experiment was over. They were also asked if they felt

any time pressure during the experiment whereupon participant three answered “yes, in the beginning”, while the three others did not feel any time pressure. The fact that participant three felt time pressure may have contributed to the increased energy use.

The experiment was successful in replicating an early-stage design context, although many other experiments or variations of the egg-drop-challenge could have highlighted the possible uses of the application better. In the chosen experiment, the work locations of the participants were very stationary. An interesting variation could have been to let the participants use all the tools available within the sensing area in the workshop, and then use the heat map to analyze the strategies of the participants. Further, reducing the available time could have facilitated more activity and energy-use. Finally, a drawback of the chosen experiment is that the participants worked in positions that resulted in occluded hands so hand tracking became difficult.

Other experiments were considered:

- Comparing different brainstorming activities
- How group size influence activity in brainstorming and how the energy of the individuals are influenced.
- How energy correlate with outcome in brainstorming
- Comparing prototyping methods. What method facilitate most energy-use
- When ideas are created in brainstorming, can there also be seen a peak in energy-use? Before or after?

Unfortunately, due to lack of participants and time, these experiments must be included in future work.

11 General Reflections from Kinect in Context

During planning and experimenting with different sensing setups in early-stage design contexts, a few challenges occurred that will be discussed in this section.

11.1 Height of Rooms

Most rooms in the university where the thesis was written have a ceiling height of up to 3 m. Even though the field of view has been increased in the Kinect v2 to 70 x 60 degrees, the available sensing area soon becomes a limitation. For ceilings at 3 m height, this equals a 4.2 x 3.5 m sensing area at floor level. At the head level of a person 180 cm tall, this equals a sensing area of only 1.7 x 1.4 m. If possible, the sensor can be placed at the same height as the max recommended sensing depth by Microsoft of 4.5 m and have a floor sensing area of 6.3 x 5.2 m, but this isn't always possible due to practical reasons. Unfortunately, in many early-stage design contexts, such as prototyping, people often tend to be very mobile as they work and the sensing area may be insufficient. As discussed earlier, several Kinects could be used to expand the sensing area, but do involve a substantial overhead from merging several frames.

Figure 11.1 Setups with different heights tried in the thesis.

Heights ranging from 2.9 m to 4.4 m was tried throughout the thesis. At higher positions, each body is represented with less pixels so less information is acquired about the body. At lower positions a parallax effect is experienced. Parallax effect describes the phenomenon when the position of an object seen, in this case in the frame, is different depending on the position of the camera. Figure 11.2 shows a parallax effect for this particular case. The persons in the frame are standing up straight and only the profile from above should ideally be visible in the frame plane, still their bodies are covering larger areas of the frame because of the position of the camera.

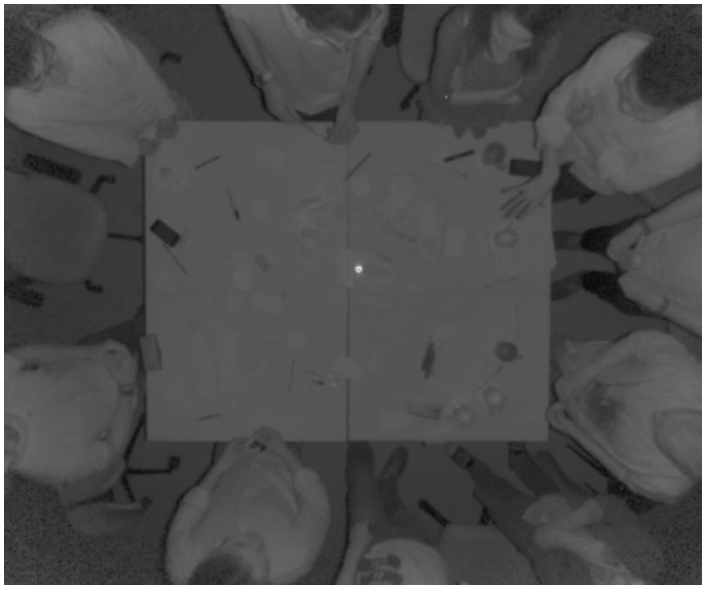


Figure 11.2 Parallax effect

The amount of parallax could be calculated from the sensor parameters and the height of the sensor. This was done in Tseng et al. (2014) and also in the thesis. However, because the resulting frame resulted in too many missing data pixels from the warping of the image, the feature was removed. It also didn't seem to affect the performance of the tracking.

11.2 Recording Sessions

The normal workflow when analyzing work sessions is to record and store the sensing sessions with Kinect Studio for further analysis or tweaking of the parameters in the tracking application. With standard surveillance equipment it is normally possible to leave the cameras on for long periods of time or let the monitoring be triggered from detection of motion. With the Kinect, this is not straight forward. Recording all the data channels, including depth, IR, audio and color for a 1 min session means storing roughly 10 GB of data, which will quickly have most hard drives running out of free space. To be able to store longer recordings and to keep up with the high data rate coming from the sensor, a thunderbolt drive with transfer speeds of up to 136

MB / sec was used together with a MacBook Pro 13” 2015, but still the CPU didn’t manage to store the data fast enough. Finally, to be able to store the sessions, only the channels for IR and depth data was used. Resulting in about 88 GB for a one hour session and the transfer speed of the thunderbolt to be able to transfer the data fast enough.

11.3 Hardware Requirements

As mentioned when comparing the Kinect v1 and v2, the hardware requirements of the Kinect v2 has increased substantially and may lead to issues, at least for some more time. The requirement of the computer having an USB 3.0 port was unexpected, as well as the recommended requirement of processing power higher than Intel I7 3.1 GHz.

11.4 USB 3.0 Cable

The supplied USB 3.0 cable and extension cable in the Kinect 2 for Windows kit are 5 m in total. The limited cable length became an issue for several setups, especially when mounting in ceilings. This issue has been extensively addressed and discussed in forums online. The official answer from Microsoft is that it is only the included cable that is supported. Still, some people have successfully used USB 3.0 cables that actively strengthens the signal throughout the cable (Microsoft Team, n.d.-b). Nevertheless, the impression from reading online is, for now, that the chances of a third-party USB 3.0 extension cable working with the Kinect 2 is relatively small. Further, the active USB 3.0 cables that could work with the Kinect 2 seems to be expensive (I don’t want to give an example before testing the cables).

11.5 Cluttered Rooms

In most locations where early-stage design is being done, the areas tend to be cluttered. Also considering limited sensing areas at low ceiling, it became early clear that a better approach than start sensing in areas already being used for early-stage design was to customize an area for sensing. In this way the interaction between people and equipment or materials can be more precisely analyzed. Further, the people tracking could be customized to the specific setup and be more precise. People detection and tracking is a very challenging task, and no approaches so far are robust enough to work in all environments.

Figure 11.3 Environments for design activities are often cluttered.

11.6 Easy to Get Started with Kinect

The Kinect SDK 2.0 provides examples covering pretty much all of the features of the Kinect v2, in both C# and C++. Very limited programming experience is needed to install an example and access the information about the bodies tracked in the Kinect SDK, such as joint-location, joint-orientation, lean vector of body, hand-states and more.

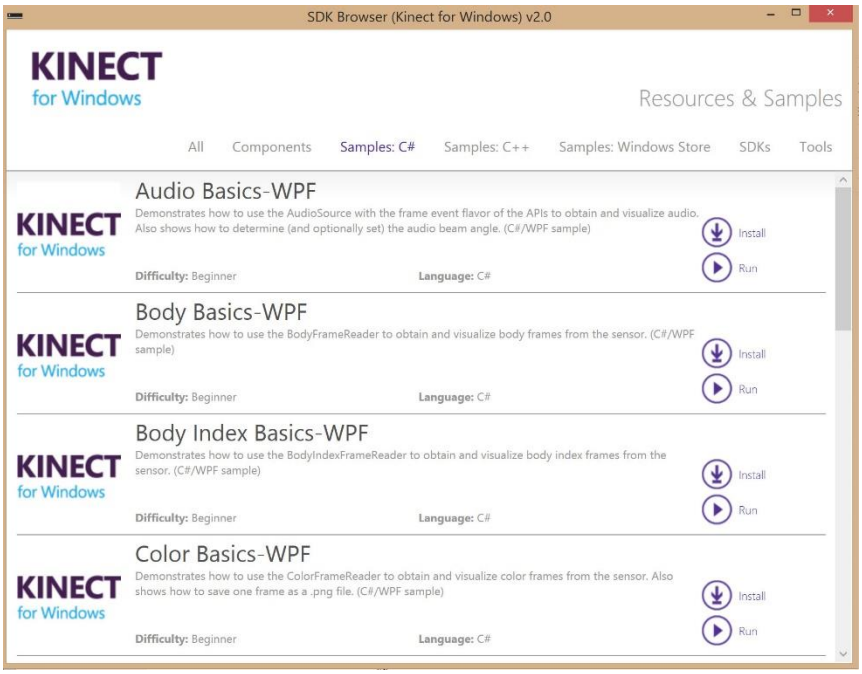


Figure 11.4 Kinect SDK Browser v2.0 is a good place to start developing an application.

12 Future Work

Future work specific for the developed body tracker could be stabilizing the tracking with Kalman filters to get more robust and consistent energy calculations. The hand tracking needs improvements, which could be done by including color data and traditional hand tracking.

The energy calculations can be smarter. For instance, the estimation of center of mass should be improved and the weight of a person could be estimated from height and size of blob of whole body. The energy estimations need to be control measured. Another heat map could also be calculated, showing not heat of locations but where in the frame energy was used.

Further, data from several Kinect v2 sensors can be combined, using more hardware and computational power. A design observatory setup can be multiple Kinects positioned high in up in a room with a clear view of the participants it is supposed to track. The number of Kinects needed is dependent on the requirements of minimizing occlusion and thereby get more stable tracking. The tracking results from the Microsoft Body Tracker, which can track 6 people very precisely, could be streamed over a network and fused together, or a tracker for the specific context can be written. In addition, object detection and tagging of locations and objects can be included to detect amount of interaction with tools and machines.

With consistent tracking of bodies and body parts, interactions between people can possibly be automatically captured and analyzed. Head orientation combined with the very capable audio recognition in the Microsoft API could tell who is talking when and to whom. By using the Kinect Gesture Builder or defining a gesture database with machine learning, interaction gestures such as pointing at someone, giving something, leaning in concentrated, leaning back relaxed, moving head confused, moving impatiently, arms crossed passively, rhythmic movements and more could be detected and possibly automatically analyzed. Energy estimates can be combined with the gestures and real-time energy estimates can be used to explore how the occurrences of creativity relate to energy use, and what type of energy use.

The audio capabilities of the Kinect v2 was not thoroughly tested in this thesis. Still, the research done indicate that speech recognition in combination with the ability to locate the source of the voice can greatly automate the cumbersome process of transcribing design sessions, or at least tag locations in the session where specific words are said. Future work involves exploring these capabilities.

Many research questions for future work that an automated design observatory could provide contributions to occurred throughout the thesis:

- *How do the amount of energy used relate to the quantity and quality of the outcome of creative sessions?*
- *How do the timing of ideas relate to energy-use?*
- *What design activities facilitate more movement?*
- *How can the interior of a room influence energy-use?*
- *How do group sizes affect individual energy-use?*
- *At what group-sizes does subgrouping start to happen?*
- *How does activity and energy-use evolve with length of sessions?*
- *How does time pressure influence energy-use?*
- *From gaze, speech patterns, gestures and pose – how much do participants pay attention when other participants are talking?*
- *What prototyping methods facilitate interaction?*
- *How is the group dynamic influenced participants are given real-time feedback on how much they are talking compared to rest of the group*

13 Conclusion

The goals of this thesis were to evaluate the Kinect v1 and Kinect v2 sensors, their potential contribution to the future of design observation and how their 3D sensing capabilities can be used to quantify human activity in design activities.

The thesis started out with reviewing current literature on design observation and non-intrusive sensing. Further, a thorough review of the Kinect v1 and v2 followed – the two sensors where compared and possibilities and limitations from a design observation perspective were highlighted. Possible observation setups with the Kinects were considered, and a setup using a Kinect v2 with a top-view of the participants was chosen. Literature relevant to achieving successful observation for the chosen approach was reviewed – existing research on human tracking in depth data from top-view as well as research on calculating energy-use from human movement. A proof-of-concept tracking application was developed for the setup. The application tracks participants, estimates mechanical energy-use real-time and creates heat maps from their movements. The tracking application was tested in a prototype of a design experiment of a realistic early-stage design context. Further, generally challenging scenarios for sensing with the Kinect were discussed, as well as issues to be aware of when using the Kinect v2 for design observation. Finally, ideas for future work were presented.

Interpreting human behavior with software can automate design observation, and using depth data to achieve this solves many of the limitations from traditional RGB data. The depth data from the Kinect is unaffected by illumination changes and color. In addition, 3D depth data greatly simplifies the task of differentiating between background and foreground as well as allowing the tracking algorithms to use the naturally characteristic 3D shapes of humans in the calculations.

In the new Kinect v2, a different depth sensing technology is used than in the old Kinect v1. The depth sensing technology called Indirect Time of Flight gives the Kinect v2 sensor more accurate depth measurements than in the predecessor. As the error in the depth measurements increases quadratic with increasing distance in the old sensor, the error in the new sensor increases linearly. Indirect Time of Flight also enables the Kinect v2 to provide a higher resolution in the depth data. The higher resolution and increased accuracy makes the Kinect v2 an interesting sensor for sensing more subtle movements, such as facial expressions and hand gestures. The Software Development Kit (SDK) provided with the Kinect v2 is also very much improved. The main disadvantages with the new Kinect v2 are the challenges met when trying

to combine data from several sensors. While the old SDKs supported multiple Kinect v1 sensors, only one Kinect v2 is supported in the new. Even though the Kinect v2 has a wider field of view, this may limit its use. Still, there are ways to utilize multiple Kinects, but so far more hardware and tweaks are necessary.

The human tracking software developed for the Kinect v2 by Microsoft has also been improved in the SDK 2.0. This body tracker is the current state-of-the-art for tracking humans from a frontal view. Even though the Microsoft body tracker is very capable, it was not used in the proof of concept application. I wanted the proof of concept application to be as non-intrusive as possible, without putting any constraints on the working position of the people in the scene. I also wanted the application to be able to capture all movements in the sensing area. For this to be achieved, occlusion needed to be minimized. Therefore a setup with one Kinect v2 mounted in the ceiling facing straight down was chosen. The option of using multiple Kinects was considered, however, for the scope of the thesis, it was not feasible to acquire hardware for two Kinect v2s. Two Kinect v1s could have been used, but the higher specifications and performance of the new Kinect v2s made it the most desirable choice.

A prototype of an experiment was developed to test the developed application and the observation setup. The prototype is a variation of the egg-drop-challenge where the participants are supposed to build a way to protect their egg from braking with the materials provided when dropped from increasing heights. The results showed that the application was able to track the participants real-time and calculate precise heat maps of their movements. The mechanical energy used by the participants was also estimated real-time, although with a few sources of error. While working, the participants often leaned over the table, which resulted in occluded hands that could not be tracked and had to be excluded from the energy calculations. Further, the energy calculations makes some assumptions about the distribution of mass in a person's body that leads to inaccuracies. Still, research indicate that with precise tracking of body segments, the calculated mechanical energy-use of each body segment is sufficient to estimate total energy-use.

Some especially challenging scenarios and sources of error for the Kinect sensors became evident throughout the thesis. Reflective and very light absorbing materials may result in the sensor not being able to calculate a depth measurement correctly or not at all. The depth measurements around the edges of an object usually become inaccurate due to a phenomenon called "flying pixels". Human trackers in general, struggles with separating humans that are positioned very close to each other. Lastly, the outer regions of the sensing area contain less

valid and more inaccurate depth measurements than the center region. This is because the infrared beams that are used to calculate the depths have to travel further in the outer regions.

For the more specific application of using Kinect v2s in a design observatory, some issues are worth being aware of. The sensing area and the sensing depths of the Kinects are limited. Careful placement and much space is necessary if bigger areas are to be covered. It is sometimes desired that sensing sessions are recorded for later analysis. The data rate produced by the Kinect sensor could potentially be enormous depending on how many of the data streams that are used. The requirement of having one USB 3.0 host controller for each sensor is final. The host controller needs to reserve the high bandwidth of the sensor. The cable shipped with the Kinect v2 for windows is 5 m. As of now, not that many extension cables seem to work. Apart from these issues, the software available from Microsoft contains loads of examples and the API is very well documented. Very little time and effort is needed to start sensing and prototyping setups with the new Kinect sensor.

Future work involves combining several Kinect v2s in a permanent design observatory. With careful placement of the sensors to minimize occlusion and the necessary hardware, very precise tracking of participants in design activities is possible. More sophisticated energy estimates can be made from the tracking of individual body segments. How a database of gestures specific for design contexts can be defined and automatically detected and analyzed should be explored. Tools can easily be tagged and the interaction with the participants can be measured. Finally, the directional audio and speech recognition capabilities of the Kinect v2 sensor should be explored and potentially be used to automatically transcribe design sessions or be combined with gestures to start automatically analyzing basic interaction between participants.

14 Reflections on Process

The start of the thesis semester was a bit hectic. The startup had been postponed until March because of an intense pre-master project at CERN, which also meant that the pre-master was about another topic than the thesis and could not be used as foundation. Looking back, I would have prioritized spending more time before the startup of my thesis narrowing down the scope. Much time was spent in the beginning of the thesis speculating in what was most interesting to sense and if it was feasible to accomplish in the thesis. I eventually realized many of the questions I had been trying to find an answer to could only be answered by starting developing the sensing.

The thesis was written at the Sino-Finnish Center (SFC) at Tongji University in Shanghai. The SFC is a joint effort of Tongji University and Aalto University in Finland, and is part of the Design Factory Network originating from the Aalto University. The ambition of the SFC is to bring students, companies and researchers from different disciplines and cultures together. I learned a lot during my stay at the SFC. Aside from learning a great deal about the research areas relevant to my thesis and getting experience in planning and executing a long-time project, I got first-hand experience with a culture very different from the Norwegian.

Everything takes more time in China, and the six extra weeks we get when doing our thesis abroad was very welcome. Things that needed to be taken care of in the beginning of the stay, such as medical examinations, registration at school and police office, residence permit, opening bank accounts and getting sim-card, all made the first month of the thesis more hectic. In China, you never know what you get and there is most of the time a problem with something. In addition to dealing with China when taking care of all the required start up hassle, hardware for the thesis eventually needed to be purchased – the Kinects, a computer that could run the Kinect v2 and thunder-drive storage. Purchases such as these are trivial in Norway, yet in China, 90% of what you buy online are fake and you never know what. Websites are in Chinese, with almost unusable translator tools, and very few speak English in stores. By default, all internet access are restricted. A VPN service can be used, for instance for all Google services, but the continuous battle between the Chinese firewalls and the VPN services makes the VPN slow and unpredictable. Through NTNU, we are provided with access to publications in several useful research databases such as Scopus, ScienceDirect and IEEE. Unfortunately, from the middle of April until June, the VPN to NTNU was blocked. Most of the time the publications could be

found by using another VPN and Google, but the process of finding research was significantly slowed down.

A project plan was created at the beginning of the thesis. The plan was to first do an explorative phase where early-stage design activities would be observed, to get a sense of what activities and parameters would be interesting to measure. The SFC had started thinking about setting up a group work room for monitoring before I arrived, and it seemed natural for me to contribute to that. Six surveillance cameras were mounted, wired to a close-by monitoring room and hooked up to a screen (Figure 52, 53). In retrospect, I realize that too much time was used on this. The goal of the monitoring setup was to observe early-stage design activities, which showed itself challenging. Barely any design activities were happening at the SFC throughout the semester, and even less in the observed room. The room had originally been dedicated to one of the project courses at the SFC. However, due to the SFC being located far from the main campus and lacking supervision in the course, the projects in the course had ceased activity. Getting more people to use the observed room was tried by hanging up posters and notes. Still, in the beginning of May it could be seen from motion detected recordings that the room only had been used 3 hours in total during the whole April. A much more efficient way of exploring design activities would have been to just use a handheld camera and go to the places where early-stage design was happening, perhaps at other locations on campus.

Figure 14.1 Some of the surveillance cameras mounted. The one up to the right could be remote controlled.

Figure 14.2 The monitoring room with monitoring setup.

It was planned to do more experiments with the Kinect earlier. Nevertheless, as many have experienced before me, estimating how much time implementation of software takes is a challenging task. Process-wise it would have been interesting to try to adopt ideas from agile software development methodology such as SCRUM and Minimum Viable Product, and through that have been able to have tracking software with basic functionalities ready earlier which could shorten the iteration time for developing the sensing setup.

14.1 Sino-Finnish Center

There were very few design projects happening at the SFC, and in general very low activity. The few students that were there outside lectures, where mostly doing homework in other courses. A couple companies had their offices there, but creative sessions or prototyping was very rarely done. A contribution to the low student activity was the fact that most of the rooms for group activities at the SFC was locked. The students could ask the staff to open the rooms, but it was rarely done, probably because they didn't know they could. Further, the university policy said that a responsible person needed to be present at the SFC at all opening hours, resulting in the place closing every evening at 21.00 and in weekends 17.00-18.00, which limited the flexibility of the students' work hours.

The SFC has great potential as a platform for intercultural interaction. Unfortunately, they haven't been very successful yet, and during my stay there some pain points have become clear. Everything is very unpredictable at the SFC. It's not clear whether courses will take place until far into the semester, there were incidences of classes being cancelled only hours before scheduled time, presentations change day the same morning as the day they were scheduled without all participants being informed, and student projects not being followed up by the supervisors and left to themselves. Also, the SFC's goal of being a facilitator for interaction between cultures has much potential for improvement. One example is that the official language of the SFC is English, still, many classes and workshops are held in Chinese which do not give a welcoming and international impression for foreigners.

There are many very capable people connected to the SFC that could have easily made the place more successful in very short time. However, that will not happen when the relevant people continue to be dishonest towards themselves and outwards to others about the reality and the real challenges. Which is also an observation about Shanghai in general, albeit from my limited time there (5 months). The obvious focus on façade over quality and integrity of the main activities in the SFC is a challenge. I spent most of my days working at the SFC, and during my

stay there I got the feeling there were more guided sightseeing tours showing of the creative interior and bragging about how successful the place was than actual students learning and interacting there. If you hold too many guided tours and tell all visitors that everything is perfect, you eventually start believing it.

My advice is to further focus and spread the strategy and core values of the SFC. Especially among the staff and lecturers. It should be clear to the students that it is a place for them, a place where they are welcome and belong. A Design Factory should contain a self-run student community and this is only possible if the students are given more freedom and trust. Also, procedures for including new students, both international and Chinese, in the routines and activities at the SFC should be developed.

The culture difference between Scandinavia and China is major. The SFC is so far no exception.

15 References

Algazi, V. R., Avendano, C., & Duda, R. O. (2001). Estimation of a Spherical-Head Model from Anthropometry. *Journal of the Audio Engineering Society*, *49*(6), 472–479.

Amon, C., & Fuhrman, F. (2014). Evaluation of the Spatial Resolution Accuracy of the Face Tracking System for Kinect for Windows v1 and v2. Presented at the 6th Congress of Alps-Adria Acoustics Assosiation.

Asteriadis, S., Chatzitofis, A., Zarpalas, D., Alexiadis, D. S., & Daras, P. (2013). Estimating Human Motion from Multiple Kinect Sensors. In *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications* (pp. 3:1–3:6). New York, NY, USA: ACM. http://doi.org/10.1145/2466715.2466727

Basu, S., Choudhury, T., Clarkson, B., & Pentl, A. (2001). Towards measuring human interactions in conversational settings. In *in IEEE Int'l Workshop on Cues in Communication (CUES 2001*.

Bauer, S., Wasza, J., Haase, S., Marosi, N., & Hornegger, J. (2011). Multi-modal surface registration for markerless initial patient setup in radiation therapy using microsoft's Kinect sensor. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (pp. 1175–1181). http://doi.org/10.1109/ICCVW.2011.6130383

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. http://doi.org/10.1023/A:1010933404324

Brekelmans, J. (2014, September 7). Multi Kinect v2. Retrieved from http://brekel.com/multikinectv2/

Brisswalter, J., Collardeau, M., & René, A. (2012). Effects of Acute Physical Exercise Characteristics on Cognitive Performance. *Sports Medicine*, *32*(9), 555–566. http://doi.org/10.2165/00007256-200232090-00002

Budiu, M., Shotton, J., Murray, D. G., & Finocchio, M. (2011). Parallelizing the training of the Kinect body parts labeling algorithm. *Big Learning: Algorithms, Systems and Tools for Learning at Scale*, 1–6.

Butkiewicz, T. (2014). Low-cost coastal mapping using Kinect v2 time-of-flight cameras. In *Oceans - St. John's, 2014* (pp. 1–9). http://doi.org/10.1109/OCEANS.2014.7003084

Butler, D. A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., & Kim, D. (2012). Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1933–1936). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2208335

Carrizosa, K., Eris, Ö., Milne, A., & Mabogunje, A. (2002). Building the Design Observatory: a Core Instrument for Design Research. *DS 30: Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005* (Vol. 1, pp. 886–893 vol. 1). http://doi.org/10.1109/CVPR.2005.177

del-Blanco, C. R., Mantecón, T., Camplani, M., Jaureguizar, F., Salgado, L., & García, N. (2014). Foreground Segmentation in Depth Imagery Using Depth and Spatial Dynamic Models for Video Surveillance Applications. *Sensors (Basel, Switzerland)*, *14*(2), 1961–1987. http://doi.org/10.3390/s140201961

de Leva, P. (1996). Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of Biomechanics*, *29*(9), 1223–1230.

Dillencourt, M. B., Samet, H., & Tamminen, M. (1992). A General Approach to Connected-component Labeling for Arbitrary Image Representations. *J. ACM*, *39*(2), 253–280. http://doi.org/10.1145/128749.128750

Dinar, M., Shah, J. J., Cagan, J., Leifer, L., Linsey, J., Smith, S. M., & Hernandez, N. V. (2015). Empirical Studies of Designer Thinking: Past, Present, and Future. *Journal of Mechanical Design*, *137*(2), 021101–021101. http://doi.org/10.1115/1.4029025

Dutta, T. (2012). Evaluation of the Kinect™ sensor for 3-D kinematic measurement in the workplace. *Applied Ergonomics*, *43*(4), 645–649. http://doi.org/10.1016/j.apergo.2011.09.011

E3: Microsoft shows off gesture control technology for Xbox 360. (2009, June 1). Retrieved from http://latimesblogs.latimes.com/technology/2009/06/microsofte3.html

Fernandez-Sanchez, E., Diaz, J., & Ros, E. (2013). Background Subtraction Based on Color and Depth Using Active Sensors. *Sensors*, *13*(7), 8895–8915. http://doi.org/10.3390/s130708895

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. http://doi.org/10.1006/jcss.1997.1504

Galna, B., Barry, G., Jackson, D., Mhiripiri, D., Olivier, P., & Rochester, L. (2014). Accuracy of the Microsoft Kinect sensor for measuring movement in people with Parkinson's disease. *Gait & Posture*, *39*(4), 1062–1068. http://doi.org/10.1016/j.gaitpost.2014.01.008

Girshick, R., Shotton, J., Kohli, P., Criminisi, A., & Fitzgibbon, A. (n.d.). *Efficient Regression of General-Activity Human Poses from Depth Images*.

Gonzalez-Jorge, H., Rodríguez-Gonzálvez, P., Martínez-Sánchez, J., González-Aguilera, D., Arias, P., Gesto, M., & Díaz-Vilariño, L. (2015). Metrological comparison between Kinect I and Kinect II sensors. *Measurement, 70*, 21–26. http://doi.org/10.1016/j.measurement.2015.03.042

González-Jorge, H., Zancajo, S., González-Aguilera, D., & Arias, P. (2015). Application of Kinect Gaming Sensor in Forensic Science. *Journal of Forensic Sciences*, *60*(1), 206–211. http://doi.org/10.1111/1556-4029.12565

Gordon, G., Darrell, T., Harville, M., & Woodfill, J. (1999). Background estimation and removal based on range and color. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (Vol. 2). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=784721

Greff, K. (2012). A COMPARISON BETWEEN BACKGROUND SUBTRACTION ALGORITHMS USING A CONSUMER DEPTH CAMERA. In *VISAPP* (pp. 431–436). SciTePress - Science and and Technology Publications. http://doi.org/10.5220/0003849104310436

Huang, T. S., Yang, G. J., & Tang, G. Y. (1979). A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, *27*(1), 13–18.

Jia, W., Yi, W.-J., Saniie, J., & Oruklu, E. (2012). 3D image reconstruction and human body tracking using stereo vision and Kinect technology. In *2012 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 1–4). http://doi.org/10.1109/EIT.2012.6220732

Kalman, R. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)), 35–45.

Kępski, M., & Kwolek, B. (2014). Person Detection and Head Tracking to Detect Falls in Depth Maps. In *Computer Vision and Graphics* (pp. 324–331). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-319-11331-9_39

Kim, K., Chalidabhongse, T. H., Harwood, D., & Davis, L. (2005). Real-time Foreground-background Segmentation Using Codebook Model. *Real-Time Imaging*, *11*(3), 172–185. http://doi.org/10.1016/j.rti.2004.12.004

Kinect. (n.d.). Retrieved August 30, 2015, from https://en.wikipedia.org/wiki/Kinect

Kinect for Windows. (n.d.). Retrieved May 19, 2015, from https://www.microsoft.com/en-us/kinectforwindows/

Kohli, P., & Shotton, J. (2013). Key developments in human pose estimation for kinect. In *Consumer Depth Cameras for Computer Vision* (pp. 63–70). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4471-4640-7_4

Kolb, A. (2009). *Dynamic 3D Imaging: DAGM 2009 Workshop, Dyn3D 2009, Jena, Germany, September 9, 2009, Proceedings*. Springer Science & Business Media.

Kunz, A., Alavi, A., & Sinn, P. (2014). Integrating Pointing Gesture Detection for Enhancing Brainstorming Meetings Using Kinect and PixelSense. *Procedia CIRP*, *25*, 205–212. http://doi.org/10.1016/j.procir.2014.10.031

Lachat, E., Macher, H., Mittet, M.-A., Landes, T., & Grussenmeyer, P. (2015). FIRST EXPERIENCES WITH KINECT V2 SENSOR FOR CLOSE RANGE 3D MODELLING. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XL-5/W4*, 93–100. http://doi.org/10.5194/isprsarchives-XL-5-W4-93-2015

Lepetit, V., Lagger, P., & Fua, P. (2005). Randomized trees for real-time keypoint recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005* (Vol. 2, pp. 775–781 vol. 2). http://doi.org/10.1109/CVPR.2005.288

Lienhart, R., Kuranov, A., & Pisarevsky, V. (2003). Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In B. Michaelis & G. Krell (Eds.), *Pattern Recognition* (pp. 297–304). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-45243-0_39

Liu, Z., Tang, S., Qin, H., & Bu, S. (2012). Evaluating User's Energy Consumption Using Kinect Based Skeleton Tracking. In *Proceedings of the 20th ACM International Conference on Multimedia* (pp. 1373–1374). New York, NY, USA: ACM. http://doi.org/10.1145/2393347.2396491

Li, W., Zhang, Z., & Liu, Z. (2010). Action recognition based on a bag of 3d points. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on* (pp. 9–14). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5543273

McCowan, I., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., & Zhang, D. (2005). Automatic analysis of multimodal group actions in meetings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(3), 305–317. http://doi.org/10.1109/TPAMI.2005.49

Microsoft Team. (n.d.-a). Kinect for Windows SDK 2.0. Retrieved from https://msdn.microsoft.com/nb-no/library/dn799271.aspx

Microsoft Team. (n.d.-b). Which USB 3 extenders are Kinect v2 compatible? Retrieved from https://social.msdn.microsoft.com/Forums/en-US/17f76f3c-810e-4b4f-a94f-2c706f5a6c49/which-usb-3-extenders-are-kinect-v2-compatible?forum=kinectv2sdk

Minneman, S., Harrison, S., Janssen, B., Kurtenbach, G., Moran, T., Smith, I., & van Melle, B. (1995). A Confederation of Tools for Capturing and Accessing Collaborative Activity. In *Proceedings of the Third ACM International Conference on Multimedia* (pp. 523–534). New York, NY, USA: ACM. http://doi.org/10.1145/217279.215316

Min Sun, Kohli, P., & Shotton, J. (2012). Conditional regression forests for human pose estimation (pp. 3394–3401). IEEE. http://doi.org/10.1109/CVPR.2012.6248079

Nakamura, Y. (2012). Human Sensing. In T. Ishida (Ed.), *Field Informatics* (pp. 39–53). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-29006-0_3

Nathan, D., Huynh, D. Q., Rubenson, J., & Rosenberg, M. (2015). Estimating Physical Activity Energy Expenditure with the Kinect Sensor in an Exergaming Environment. *PLoS ONE*, *10*(5). http://doi.org/10.1371/journal.pone.0127113

Nguyen, D. T., Li, W., & Ogunbona, P. (2009). A part-based template matching method for multi-view human detection. In *Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference* (pp. 357–362). http://doi.org/10.1109/IVCNZ.2009.5378380

Nguyen, V.-T., Vu, H., & Tran, T.-H. (2015). An Efficient Combination of RGB and Depth for Background Subtraction. In Q. A. Dang, X. H. Nguyen, H. B. Le, V. H. Nguyen, & V. N. Q. Bao (Eds.), *Some Current Advanced Researches on Information and Computer Science in Vietnam* (Vol. 341, pp. 49–63). Cham: Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-14633-1_4

Oosterhout, T. (n.d.). HEAD DETECTION IN STEREO DATA FOR PEOPLE COUNTING AND SEGMENTATION.

OpenKinect. (2014, June). Multiple kinects V2 tested. Retrieved from https://github.com/christiankerl/libfreenect2/issues/7#issuecomment-47422005

OpenKinect.org. (n.d.). Retrieved June 7, 2015, from http://openkinect.org/wiki/Main_Page

Oppezzo, M., & Schwartz, D. L. (2014). Give your ideas some legs: The positive effect of walking on creative thinking. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *40*(4), 1142–1152. http://doi.org/10.1037/a0036577

Perreault, S., & Hébert, P. (2007). Median filtering in constant time. *Image Processing, IEEE Transactions on*, *16*(9), 2389–2394.

Plagemann, C., Ganapathi, V., Koller, D., & Thrun, S. (2010). Real-time identification and localization of body parts from depth images. In *2010 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3108–3113). http://doi.org/10.1109/ROBOT.2010.5509559

PrimeSense Supplies 3-D-Sensing Technology to "Project Natal" for Xbox 360. (2010, April 31). Retrieved from http://news.microsoft.com/2010/03/31/primesense-supplies-3-d-sensing-technology-to-project-natal-for-xbox-360/

Qwertie. (n.d.). Head-to-head benchmark: C++ vs .NET. Retrieved from http://www.codeproject.com/Articles/212856/Head-to-head-benchmark-Csharp-vs-NET

Rainer Stiefelhagen. (2002, May 6). *Tracking and Modeling Focus of Attention in Meetings*. Universitat Karlsruhe.

Ren, Z., Yuan, J., & Zhang, Z. (2011). Robust Hand Gesture Recognition Based on Finger-earth Mover's Distance with a Commodity Depth Camera. In *Proceedings of the 19th ACM International Conference on Multimedia* (pp. 1093–1096). New York, NY, USA: ACM. http://doi.org/10.1145/2072298.2071946

Santhanam, T., Sumathi, C. P., & Gomathi, S. (2012). A Survey of Techniques for Human Detection in Static Images. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology* (pp. 328–336). New York, NY, USA: ACM. http://doi.org/10.1145/2393216.2393272

Scopus. (n.d.). Retrieved June 8, 2015, from http://www.scopus.com/

Seer, S., Brändle, N., & Ratti, C. (2014a). Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research Part C: Emerging Technologies*, *48*, 212–228. http://doi.org/10.1016/j.trc.2014.08.012

Seer, S., Brändle, N., & Ratti, C. (2014b). Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research Part C: Emerging Technologies*, *48*, 212–228. http://doi.org/10.1016/j.trc.2014.08.012

Sharp, T. (2012). The Vitruvian Manifold: Inferring Dense Correspondences for One-shot Human Pose Estimation. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 103–110). Washington, DC, USA: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=2354409.2354668

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., … Blake, A. (2013). Real-Time Human Pose Recognition in Parts from Single Depth Images. In R. Cipolla, S. Battiato, & G. M. Farinella (Eds.), *Machine Learning for Computer Vision* (pp. 119–135). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-28661-2_5

Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., … Blake, A. (2013). Efficient Human Pose Estimation from Single Depth Images. In A. Criminisi & J. Shotton (Eds.), *Decision Forests for Computer Vision and Medical Image Analysis*

(pp. 175–192). Springer London. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4471-4929-3_13

Stone, E. E., & Skubic, M. (2011). Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)* (pp. 71–77).

Surie, D., Lindgren, H., & Qureshi, A. (2013). Kitchen AS-A-PAL: Exploring Smart Objects as Containers, Surfaces and Actuators. In A. van Berlo, K. Hallenborg, J. M. C. Rodríguez, D. I. Tapia, & P. Novais (Eds.), *Ambient Intelligence - Software and Applications* (pp. 171–178). Springer International Publishing. Retrieved from http://link.springer.com/chapter/10.1007/978-3-319-00566-9_22

Surie, D., Partonia, S., & Lindgren, H. (2013). Human Sensing Using Computer Vision for Personalized Smart Spaces. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)* (pp. 487–494). http://doi.org/10.1109/UIC-ATC.2013.24

Tang, A., Lu, K., Wang, Y., Huang, J., & Li, H. (2015). A Real-Time Hand Posture Recognition System Using Deep Neural Networks. *ACM Trans. Intell. Syst. Technol.*, *6*(2), 21:1–21:23. http://doi.org/10.1145/2735952

Tang, J. C. (1989). Listing, Drawing and Gesturing in Design: A Study of the Use of Shared Workspaces by Design Teams. Retrieved August 29, 2015, from http://cumincad.scix.net/cgi-bin/works/Show?1b88

Tang, J. C., & Leifer, L. J. (1988). A framework for understanding the workspace activity of design teams. In *Proceedings of the 1988 ACM conference on Computer-supported*

cooperative work (pp. 244–249). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=62285

Tang, J. C., & Leifer, L. J. (1991). An observational methodology for studying group design activity. *Research in Engineering Design*, *2*(4), 209–219. http://doi.org/10.1007/BF01579218

TEIXEIRA, T., DUBLON, G., & SAVVIDES, A. (2010). A Survey of Human-Sensing: Methods for Detecting Presence, Count, Location, Track, and Identity. *ENALAB Technical Report*. Retrieved from http://thiagot.com/papers/teixeira_techrep10_survey_of_human_sensing.pdf

Tian, Q., Zhou, B., Zhao, W., Wei, Y., & Fei, W. (2013). Human Detection using HOG Features of Head and Shoulder Based on Depth Map. *Journal of Software*, *8*(9). http://doi.org/10.4304/jsw.8.9.2223-2230

Tomporowski, P. D. (2003). Effects of acute bouts of exercise on cognition. *Acta Psychologica*, *112*(3), 297–324. http://doi.org/10.1016/S0001-6918(02)00134-8

Törlind, P., Sonalkar, N., Bergström, M., Blanco, E., Hicks, B., & McAlpine, H. (2009). Lessons Learned and Future Challenges for Design Observatory Research. *DS 58-2: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 2, Design Theory and Research Methodology, Palo Alto, CA, USA, 24.-27.08.2009*.

Torres, R., Huerta, M., Clotet, R., González, R., Sánchez, L. E., Rivas, D., & Erazo, M. (2015). A kinect based approach to assist in the diagnosis and quantification of parkinson's disease (Vol. 49, pp. 461–464). Presented at the IFMBE Proceedings. http://doi.org/10.1007/978-3-319-13117-7_118

Tseng, T.-E., Liu, A.-S., Hsiao, P.-H., Huang, C.-M., & Fu, L.-C. (2014). Real-time people detection and tracking for indoor surveillance using multiple top-view depth cameras.

In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)* (pp. 4077–4082). http://doi.org/10.1109/IROS.2014.6943136

Van de Walle, P., Hallemans, A., Schwartz, M., Truijen, S., Gosselink, R., & Desloovere, K. (2012). Mechanical energy estimation during walking: Validity and sensitivity in typical gait and in children with cerebral palsy. *Gait & Posture*, *35*(2), 231–237. http://doi.org/10.1016/j.gaitpost.2011.09.012

Vangos Pterneas. (n.d.). How to use Kinect HD Face. Retrieved from http://www.codeproject.com/Articles/998379/How-to-use-Kinect-HD-Face

Velardo, C., & Dugelay, J.-L. (2011). Real time extraction of body soft biometric from 3D videos (p. 781). ACM Press. http://doi.org/10.1145/2072298.2072454

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001* (Vol. 1, pp. I–511–I–518 vol.1). http://doi.org/10.1109/CVPR.2001.990517

Vladimir Zatsiorsky, & Boris Prilutsky. (n.d.). *Biomechanics of Skeletal Muscles*. Human Kinetics.

When will the driver support multiple Kinect 2's per computer? (2014, September 20). Retrieved from https://social.msdn.microsoft.com/Forums/en-US/294c89ca-c947-4e5c-8f00-af66417ea12e/when-will-the-driver-support-multiple-kinect-2s-per-computer?forum=kinectv2sdk

Williams, K. R. (1985). The relationship between mechanical and physiological energy estimates. *Medicine and Science in Sports and Exercise*, *17*(3), 317–325.

Wilson, A. (2015). *RoomAlive Toolkit*. Microsoft. Retrieved from http://research.microsoft.com/en-us/projects/roomalivetoolkit/

Yang, K., Wei, B., Wang, Q., Ren, X., Xu, Y., & Liu, H. (2014). A 3-D Depth Information Based Human Motion Pose Tracking Algorithms. *Sensors & Transducers (1726-5479)*, *174*(7). Retrieved from http://www.sensorsportal.com/HTML/DIGEST/july_2014/Vol_174/P_2208.pdf

Yang, L., Zhang, L., Dong, H., Alelaiwi, A., & Saddik, A. E. (2015). Evaluating and improving the depth accuracy of Kinect for Windows v2. *IEEE Sensors Journal*, *15*(8), 4275–4285. http://doi.org/10.1109/JSEN.2015.2416651

Zhang, X., Yan, J., Feng, S., Lei, Z., Yi, D., & Li, S. Z. (2012). Water Filling: Unsupervised People Counting via Vertical Kinect Sensor. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance (AVSS)* (pp. 215–220). http://doi.org/10.1109/AVSS.2012.82

Zhang, Z. (2012). Microsoft Kinect Sensor and Its Effect. *IEEE MultiMedia*, *19*(2), 4–10. http://doi.org/10.1109/MMUL.2012.24

Zhou, Q., & Aggarwal, J. K. (2001). Tracking and classifying moving objects from video. In *Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance* (pp. 46–54). Hawaii, USA.

Zhu, L., & Wong, K.-H. (2013). Human Tracking and Counting Using the KINECT Range Sensor Based on Adaboost and Kalman Filter. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, B. Li, F. Porikli, … D. Gotz (Eds.), *Advances in Visual Computing* (pp. 582–591). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-41939-3_57

16 Appendix

16.1 Dropbox folder

URL:
https://www.dropbox.com/sh/zmmljibbnk1ycx4/AAC5nwlxHsAM2ICH3GzRSgY_a?dl=0

Content	Explanation
BugRoomPosters [subfolder]	Posters made to recruit and inform people about the observation room.
Diagnostics [subfolder]	Raw .xlsx files from various diagnostics.
Photos [subfolder]	Photos taken throughout the thesis.
FourMetersSitting1.xef	Kinect recording of a person walking into the frame and sitting by a table with the Kinect mounted at 4 meters height.
LeifErikBjoerkliA3.pdf	Mandatory A3 poster made in the beginning of the project describing the thesis.
RiskAssessment.pdf	Mandatory risk assessment form.
ThesisContract.pdf	The thesis contract with supervisor.

16.2 Recordings

Due to the size of the recordings, ranging from 2-90 GB and about 200GB in total, the recordings where copied to an external hard-drive and given to my supervisor Martin Steinert. One recording is available in the Dropbox folder.

16.3 Tracking application on GitHub

The tracking application can be cloned from the following GitHub repository:

https://github.com/leiferikbjorkli/Kinect2TrackingTopView.git