Low-Bandwidth Transmission of Body Scan Using Skeletal Animation

by

Jason Bradley Nel

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electrical and Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. H.A. Engelbrecht

December 2020

Plagiarism Declaration

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. I also understand that direct translations are plagiarism.
4. Accordingly all quotations and contributions from any source whatsoever (including the Internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Name:	JasonBradleyNel
Date:	2019/12/12

Abstract

Low-Bandwidth Transmission of Body Scan Using Skeletal Animation

J.B. Nel

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (E & E)

December 2020

With the recent heightened commercial availability of virtual reality head-mounted displays, methods of creating content for such systems are now becoming a relevant concern. Computer generated content is currently widely used for such purposes, as it can easily be geared towards virtual environments viewed on virtual reality displays. Recording content from the real world, however, still poses many challenges.

Two common problems in recording video for virtual reality are that the capturing process can require extensive hardware to completely capture a full scene and that large amounts of bandwidth are required to transmit the data generated by 3D video capture hardware.

A system is created for this study which attempts to address and avoid those problems, in exchange for a system which is less accurate by means of recreating the experience rather than directly relaying it. As static environments can be captured through other methods, the system records only a human actor.

The purpose of this study is to determine whether the animation of a body scan produced by this system can still be an immersive virtual reality experience despite the reduction of accuracy that may be necessary to achieve a low-cost, low-bandwidth solution.

A single off-the-shelf 3D camera, the Xbox Kinect, is used to capture 3D and skeleton data, as well as other useful data sources such as RGB video and audio.

Using only this device allows an easy set-up, at a low cost, while the software approach of animating a body scan with skeleton data allows for low bandwidth transmission over a network.

The system created for this study captures, processes, and transmits a body scan over a network in real time. The resultant body scan is then displayed in a virtual environment on a virtual reality head-mounted display.

This system achieved a data reduction of over 99% (when compared the original data or a system with similar aims) and the quality was evaluated favourably in a survey of 38 participants recruited to view a demonstration of the system.

Uittreksel

Lae-Bandwydte Oordrag van Liggaamskandering Deur Skeletale Animasie

(Low-Bandwidth Transmission of Body Scan Using Skeletal Animation)

J.B. Nel

Departement Elektriese en Elektroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (E & E)

Desember 2020

Metodes om inhoud vir kopgemonteerde virtuele realiteitskerms te skep word 'n relevante probleem as gevolg van die onlangse verhoogde kommersiële beskikbaarheid van sulke stelsels. Tans is rekenaargegenereerde inhoud wyd in gebruik vir sulke doeleindes, aangesien dit maklik te rig is op die virtuele omgewings wat op virtuele werklikheidskerms waargeneem word. Daarenteen bied die opname van inhoud uit die regte wêreld nog baie uitdagings.

Twee algemene probleme met die opname van video vir virtuele werklikheid is dat die opneemproses uitgebreide hardeware kan benodig om 'n volle toneel in geheel vas te vang en dat die oordrag van die data wat deur 3-D video-opname hardeware gegenereer word, groot hoeveelhede bandwydte benodig.

Vir hierdie studie word 'n stelsel geskep wat poog om hierdie probleme aan te spreek en te vermy, in ruil vir 'n stelsel wat minder akkuraat is deur eerder die ervaring te herskep as om dit direk oor te dra. Siende dat statiese omgewings op ander maniere vasgelê kan word, neem die stelsel slegs 'n menslike akteur op.

Die doel van die studie is om te bepaal of die animasie van ’n liggaamskandering wat deur hierdie stelsel produseer word, ondanks die vermindering van die akkuraatheid wat nodig mag wees vir ’n lae-koste, lae bandwydte oplossing, steeds ’n ingedompelde virtuele werklikheidservaring kan lewer.

’n Enkele verbruikersgraad 3D-kamera, die Xbox Kinect, is gebruik om 3D- en skelet-data, asook ander bruikbare databronne soos RGB-video en klank, op te neem.

Die gebruik van slegs hierdie toestel maak maklike opstelling teen ’n lae koste moontlik, terwyl die sagteware benadering om ’n liggaamskandering met skelet-data te animeer, lae-bandwydte oordrag oor ’n netwerk moontlik maak.

Die stelsel wat vir die studie geskep is, neem ’n liggaamskandering op, verwerk en stuur dit intyds oor ’n netwerk. Die resulterende liggaam geskandeer word dan in ’n virtuele omgewing vertoon op ’n kopgemonteerde virtuele werklikheidskerm.

Hierdie stelsel het ’n data-vermindering van meer as 99% behaal (in vergelyking met die oorspronklike data of ’n stelsel met soortgelyke doelstellings) en in ’n opname onder 38 deelnemers gewerf om ’n demonstrasie van die stelsel te aanskou, is die gehalte gunstig beoordeel.

Acknowledgements

I would like to thank the MIH Media Lab, for funding my studies and providing me with a vibrant and inspiring workplace. I would also like to thank my supervisor, Professor Herman Engelbrecht, for providing excellent guidance and getting me involved in interesting projects and opportunities.

I would like to thank my family, for their constant support, and I would like to thank Richard Barnett, for an incredible amount of guidance, advice and support over the years.

o O o
..

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The concept of virtual reality, which is by no means novel, has recently begun to experience a resurgence of popularity since interest in the technology faded in the 1990s. This is mostly the result of a sudden increase in virtual reality head-mounted displays available to the consumer market. While many of these products have seemed geared towards gaming and entertainment, their accessibility and affordability have created opportunities for virtual reality experiences to be used for other areas, such as communication and education.

As the technical challenges of VR, in both hardware and software, are overcome, it has become clear that there are still challenges present related to the content to be viewed on these devices, and how such content is created, stored, and transmitted.

1.1 Motivation

Virtual reality environments require a much higher level of content than either 2D or stereoscopic 3D video displays of comparable levels of detail. Rendering environments onto virtual reality displays requires a greater amount of graphical processing power, so that the display may update sufficiently fast to create a sense of immersion for the user. [1]

For virtual environments that are recorded rather than computer generated, there is an additional challenge in the storage and transmission of the large amount of data from which these environments are composed. Static elements of recorded virtual environments such as rooms and static objects are easily recorded and transmitted once. However, more dynamic elements — such as a human body in that environment — will most likely be in motion fairly often, and as such would have to be repeatedly updated in the playback or transmission of that element.

This study investigates an approach which attempts to reduce the amount of data required to transmit the motion of a human body into a virtual reality system, over a network, by sending a model of that body once and reanimating the model with a stream of skeletal motion data. The study aims to evaluate whether this approach, with the possible cost of a reduction in quality compared to an approach which directly transmits a virtual body scan, will still achieve a result that feels realistic to participants.

1.2 Objectives

The objectives of this study are:

- To build a system that reduces the amount of data required to store or transmit a 3D body scan for the purposes of VR, with the possible cost of a reduction of scan quality.
- To evaluate the quality of the resultant stored or transmitted scan of that system in the context of a virtual reality environment, viewed through a virtual reality head-mounted display.
- To determine whether the reduction in the quality of a body scan is acceptable when compared to the reduction in the bandwidth required for transmission that is achieved.
- To determine whether this method of transmitting body scans is viable and adaptable to better scanning technologies and faster networks.

1.3 Applications

Virtual reality currently has a range of applications for various forms of communication, education, and entertainment. In the context of these applications, this system, or one like it, would allow a user to record or transmit a virtual representation of themselves for a virtual environment, allowing viewers of the scan to see a full range of movements of the users body. Better quality scans with finer detail could more accurately and realistically represent body movements.

A wide range of virtual reality experiences could be enhanced with this type of system, which places users in those virtual environments. Any activities which involve human interaction in a physical space can be simulated, providing an enhanced experience in cases where current technologies and methods already try to approximate these actions with audio or video, and opening up the possibility of simulating experiences that have not yet been able to be recreated with current technologies and methods, due to the importance of spatial factors that are not able to be captured by 2D video.

1.3.1 Communication

Much like how audio and video calls have provided a greater deal of realism and immersion to human communications, a virtual scan would allow for substantially more engaging person-to-person interaction in a virtual environment. Body language can much more closely approach the ability it has in the real world to convey meaning, as movements can be displayed and interpreted more accurately, and virtual environments would create an illusion of a shared physical location. One example of this can be seen in a virtual reality conferencing system created by Zhang et al. [74]

1.3.2 Education

Various educational virtual experiences have been created — allowing students to immerse themselves fully in the subject matter being taught. Restrepo et al. created and analysed the effects of a telepresence learning system. This system allowed students to achieve a level of understanding of the subject matter that was similar, and in some cases better, than students in a face-to-face setting. [46] Being perceptually anchored into a learning environment with a realistic body scan, along with educators and peers, could enhance the immersiveness of the experience and increase its effectiveness as an educational tool.

1.3.3 Training

2D training videos exist for a number of physical tasks, both in the consumer space and for industry training. Virtual reality recordings used in this task could provide a clearer and more nuanced method for displaying such training, where users are able to adjust their viewing angles and distances, and more closely study details in the recording. Such a system could even be able to replace training sessions where 2D video is insufficient, and a physical presence from an instructor is usually required.

In a study by Mathur, the researcher created a training environment for medical procedures, allowing a user to identify organs and create incisions. More advanced training systems could feature multiple users, where some users are present to guide others. [38]

Stotko et al. developed a system that allows real time capture and multiple user exploration of static environments. [62] Such a system could be used for on-site training for various professions which require specialized physical training.

1.3.4 Guided Tours

Virtual tours exist in many forms. Including, for example, AR (Augmented Reality) tours that provide an informative overlay to real cities or other notable physical environments; and virtual tours which recreate historical scenes.

A game made by Herbst et al. allows for augmented reality guided city exploration, superimposing generated images onto camera footage of real cities. [15] Systems like the one created by Stotko et al. could allow for more spontaneous exploration of environments by multiple users. [62] A scanned body inserted into such environments could act as a tour guide: leading users, providing information and answering questions.

1.3.5 Art and Entertainment

Performances can be recorded in this medium, and delivered as entertainment in a variety of methods. While purely observable experiences would be an option, the nature of VR would also allow, and perhaps suggest, more interactive experiences. A performance art piece was created from the early code of the system developed for this study, where constructed skeletons were animated from Kinect second generation sensor motion data [33]. Body scans can augment such performances with realistic avatars, and create possibilities for audience interaction.

1.3.6 Virtual Sports

Motion capture data from sensors could be used to provide input for virtual sport systems as well as animate body scans of players in that world. Such a system has already been created by Raghuraman et al. for two users to play tennis using screens or projectors [45].

Using VR and inserted body scans, a convincing virtual sport experience could be created where users feel immersed and engaged in the game they are playing, with an enhanced sense of an opponent or teammate's presence in such virtual environments.

1.4 System Composition

The system of this study will require four main components. These components are:

1. Capture of the Kinect mesh and skeleton data.
2. Processing the mesh data into a body scan which is capable of being animated.
3. Transmission of the Kinect data. Once in the case of the body scan, and continuous streaming in the case of the skeleton data.
4. Displaying the body scan and animating it with the stream of skeleton data.

The resultant system will be a combination of these components allowing a user to prepare a body scan, capture an animation on one computer and transfer this animation over a network to another computer in real time.

1.5 System Specifications

For the concept put forth by this study to be useful in the context of the proposed applications, certain objective and subjective requirements must be fulfilled. With the proposed applications of the system in mind, this study aims to build a system which:

- Transmits a body scan at a lower bandwidth than that which would be required for directly transmitting the unprocessed scan data over the network.
- Performs server side processing to achieve this transmission in real time.
- Has a brief software set-up time on a reasonably powerful personal computer.
- Has a brief and uncomplicated hardware set-up.
- Does not require expensive or specialist hardware.
- Is not subjectively considered to be of substantially lower quality than the original scan.

Generally speaking, the system must achieve a certain bandwidth reduction, within the framework of the processing power of available computers and a suitably low cost for scanning hardware. The nature of these requirements will be discussed in the following sections.

1.5.1 Bandwidth

Ideally, the system would be able to transmit data over the Internet at reasonable connection speeds. As of 2018, South Africa's mean Internet speed was 6.38 Mbps. [20] A bandwidth value significantly lower than this would be optimal for this system.

1.5.2 Processing Power

While processing of data is preferable to a high bandwidth transmission, the amount of processing done should not exceed the capabilities of a fairly powerful personal computer. Additionally, once-off preliminary processing of the scan data should not take more than a few minutes, and frame-by-frame processing for animation should occur at a speed which allows for the maximum frame rate available from the sensor to be transmitted over a network in real time.

1.5.3 Cost

As a proof of concept, this system does not aim to achieve a production level of quality. The costs of the system are preferably kept low, using free software, and limiting the hardware to a reasonably powerful personal computer, a scanning device, and a mounting for that scanning device (such as a regular camera tripod.) Additionally, for viewing, the system would require a virtual reality display and a computer with sufficient graphical capabilities to power that display, as well as a method of user input.

1.5.4 Subjective Evaluation

The system must be evaluated by participants who were not involved in the creation of the system to determine to what degree the bandwidth reduction method of the system has reduced the quality of the transmitted scan. As the intended applications for this system are for the purposes of virtual reality, evaluations of the system should be conducted using a virtual reality head mounted display, so that users may evaluate the subjective qualities of the scan in the intended environment. Such an evaluation will ideally show that the method used to achieve a low bandwidth transmission of an animated body scan only results in a relatively minor reduction in the subjective quality of the scan when viewed in a virtual reality environment.

1.5.5 Virtual Reality Requirements

For an effective virtual reality experience, a virtual reality display (and the software which drives it) must be able to meet certain requirements. Abrash, a member of staff from Valve Inc. involved in the prototyping of such devices, described the requirements for virtual reality in a presentation at the 2014 Steam Dev Days conference [1] as the following:

- Wide field of view. (At least 80 degrees.)
- Adequate resolution. (At least 1080p)
- Low pixel persistence. (Lower than 3ms — to avoid blurring during head movement.)
- High refresh rate. (Greater than 60hz — 95hz was considered ideal, but not essential.)
- Global display. (Display which simultaneously updates all pixels.)
- Optics and Optical Calibration. (Which reduce the focal distance to the screen without being to heavy for a head mounted display.)

- Efficient tracking. (Millimetre accuracy, 1.5 meter ranges for motion, negligent latency.)
- Low latency. (Less than 25 ms to update image.)

While these are goals that should be met for an immersive VR system, they will not be results of the development of the system, and will be products of the choice of hardware and supporting software used in this system for virtual reality. Therefore, when choosing our hardware, it is important to choose a device that meets these requirements.

It should also be noted that while our selected scanning system will have a frame rate, and the transmission of our animations will have a latency, these are independent of the refresh-rate and image update latency listed here. For example, a scan which updates at 30Hz — a lower frame rate than the required 60Hz — would still be acceptable so long as the users view of the scene in the VR device is updating at a rate faster than 60Hz. A similar principle applies with the system and VR latencies.

1.6 Summary

This study hopes to create and investigate a system that is generally accessible — with low cost, low bandwidth usage and low processing power requirements. The system must capture, process and transmit a human body scan over a network to a VR display. This system will be considered successful if it meets bandwidth and processing time requirements, and is subjectively evaluated by users to be of reasonable quality, specifically in the context of VR.

Should this approach be successful, it can be adapted for virtual reality technologies as they improve, and provide a method which, when combined with other methods dealing with different areas in VR, can provide interesting and engaging VR experiences.

Chapter 2

Literature Review

The proposed system of this study is intended to scan a human body, and use a method to transmit a scan of that body across a network that is low-bandwidth and functions in real time. Literature was examined in the topics of 3D body scans, mesh animation techniques and network transmission of virtual reality environments and animated objects. For a qualitative analysis of the resultant system, literature on methods of evaluating virtual experiences was also examined.

2.1 3D Body Scans

Shingade et al. conducted a review of studies where systems for motion capture methods had been created, as well as a review of common devices and software used for human motion capture and depth scanning. Many of these systems used the first generation Xbox Kinect sensor and the Kinect SDK (Software Development Kit) for these scanning systems, and in the review the Kinect scored favourably compared to other proposed input devices. [52]

A system in that study, created by Tong et al., focuses on the obtaining a model of the human body from scanning. It uses three first generation Kinect sensors and a rotating platform. A user would stand on the platform and be rotated while two Kinect sensors on one side of the system would scan the upper and lower portions of the body, and a single sensor on the opposite side would scan the middle portion. These scans were then aligned and processed into a single body scan, which could be set into different poses. [64]

In another study, Kreylos et al. created a system of using two first generation Kinect sensors, located on opposite sides of the user being scanned and scanning full bodies, to create a real-time body scan which would act as an avatar in a generated virtual environment, designed for enabling scientific workflows in virtual reality. [26] Kreylos has also created an alternative library for controlling various versions of the Kinect sensor, including the second generation Kinect. [25]

Mekuria et al. also developed a system with two first generation Kinect sensors to create body scans, which they compressed and transmitted over a network to represent users in a generated virtual environment. [39, 41]

For their FreeCam system, Kuster et al. used a more substantive setup of two first generation Kinect sensors and three RGB cameras. The Grasshopper camera from Point Grey was chosen by the researchers for its superior image quality. These cameras were used to obtain a "multiple- and free-viewpoint" video system, where the viewing angle and position could be chosen by the user, and multiple views could be delivered. [27]

Li et al. use only a single first generation Kinect sensor to scan a user who rotates themselves while being scanned, thus allowing for a full scan. This scan is used in the system to create a watertight mesh (a mesh that fully encloses an inner volume) of a user in a desired pose, which the user must repeat throughout the scan. [28]

Chen et al. use a similar approach to take a scan of a rigid object, making use of the Kinect fusion functionality available in the Kinect SDK. They also use depth information to process the RGB feed from the Kinect, creating a more photo-realistic texture for the virtual scan object. [9]

Anthropometric scanning is quite a common approach to obtain error free meshes that are water-tight and have a fixed topology. Template human models are deformed to match mesh dimensions gathered from scans. This method removes the complexity of aligning uncoordinated mesh fragments and joint meshes which are not watertight, but quality and level of detail of the template used can result in scans which look computer generated rather than scanned. [51, 71, 11] Body models such as this have various applications: Lu et al. has created a system that can estimate body fat and percentage, for medical purposes. [34]

Hirshberg et al. provide a system of aligning and registering scans to body models simultaneously, allowing the system to better adapt to missing data from common scan flaws, such as stretches of the body being occluded by an arm that is between that body and the camera. [17]

From the body scanning systems examined in this review, it seems as if anthropometric scans are more commonly used in scientific applications, where dimensional accuracy is important, and that direct scans are more commonly used in visual applications (such as telepresence systems) where the user's subjective impression of the realism of the scan is important.

Additionally, the widespread use of the Kinect sensor for the systems examined in this review shows that it is a popular and effective choice for applications which require body scans.

2.2 Mesh Animation

Mesh animation is a fairly well-established practise in computer graphics, and substituting the transmission of a new mesh for every frame with operations to animate the mesh on the receiving side instead could provide a significant savings in bandwidth for the proposed system, should a small enough set of parameters for animation be found. Most established methods for mesh animation rely on matching vertices on meshes in different poses in the same animation being identified, which is simple for generated meshes and animations, but more complex for scanned meshes, and requires some form of manual or automatic identification. Alternatively, a sequence of meshes without matching vertices can be displayed in sequence to create an animation. This is more commonly used for scanned meshes. The basic aspects of these two methods will be further discussed in Section 3.4.

Alexa et al. detail a method for representing animations by their principal components, which can allow for a compression ratio of up to 1:133.28, with a degree of reduction of the quality and level of detail of the resultant animation. This system requires a pre-existing identification of matching points. [3]

Alexa et al. also present a method of representing a mesh by differential coordinates — a Laplacian mesh representation — which allows for realistic distortions of meshes caused by adjustments to the locations of points within the mesh, where points with unspecified adjustments move to accommodate the movement of points with a specified adjustment. [2] Sorkine et al. built on this method, showing how mesh deformations can deform complex shapes. Such deformations can even approximate natural movements. [58]

James et al. use these mesh representations to create an automated skinning method to display animations of semi-articulated objects. This approach is shown to deform meshes into realistic animation sequences. These models are constructed models with pre-existing identification of matching points. [23]

Li et al. use a system to fit garments to a model of a human body using Laplacian mesh representations for the garment. [30] The same team later created another system that automatically converts a full body scan into a rigged mesh, tied to a skeleton used to display the mesh in different poses. They do this by obtaining a skeleton representation for the scan and using the skeleton to split the scan into several cage meshes. The cage meshes are then deformed based on the movements of the skeleton into different poses. The mesh is then able to be deformed into different poses, which provide an approximation of how a scan of the body in that pose would appear. [29]

Based on these methods, it is clear that an animation created from a human body scan can either be created with a series of meshes, or a body scan which is processed in some way to be able to animated by the movement of an underlying skeleton. Using sequences of meshes to form animations has the advantage of working directly with the type of mesh data likely to be provided from scanning hardware, while methods involving animation using only a representation of a skeleton would require minimal data to create animations, which would be of particular use in a low-bandwidth application.

2.3 Network Transmission

In the context of a networked virtual environment, Hosseini et al. proposed an animation protocol for the transmission of large virtual environments with animations over a network. This protocol uses the BIFS (Binary Format for Scenes) standard, a part of the MPEG-4 standard. Notably, they mention the prospect of progressive approach to animation, that is: transmitting the start and end of an animation, and interpolating an estimated animation while more animation data is sent across the network. [19]

Capin et al. proposed a method for predicting intermediate frames in an animation sequence using a dead reckoning technique, which could be used to reduce the number of animation frames transmitted over a network or smooth animations where frames are substantially different. [8]

Mekuria et al. built a system for transmitting a body scan which they constructed from data from to Kinect first generation sensors over a network. They used standard lossy mesh compression techniques and achieved a compression ratio of 1:115. They created a framework to insert these scans into virtual environments over a network. [39, 41]

Stotko created a system that streams virtual environments rather than body scans, to allow for collaborative work. Such a system could be used in conjunction with body scans to completely represent a full virtual environment scan. [62]

If a reduction of the data required to animate a body scan over a network is not achieved via the method of animation, and mesh data must be streamed over the network directly, it will be necessary to employ a method to reduce the data required to do so. The method used by Mekuria et al. showed systems that were capable of creating this animation by compressing and streaming meshes over a network in real time, which would be required for the system of this study. [39, 41]

Alternatively, if a reduction of the data was achieved via the chosen method of animation, as would likely be the case with a skeletal animation systems, compression meshes would likely not be as necessary, but would remain an option for certain components of such a system.

2.4 Presence in Virtual Environments

Presence is the term used to refer to how successful the illusion of reality created by a virtual environment is. Loomis speaks of several concepts of how presence in a virtual environment is experienced. A very important factor in a users sense of presence is the concept of distal attribution — the process of identifying and conceptualizing physical objects detected by the senses as external objects. For example, when looking at a photograph of a tree, we feel a sense of presence by identifying that object as a tree and connecting it with our conceptual understanding of trees which we have experienced in the real world. In contrast, an image of random colours and shapes which we do not link to any particular concept will not create a similar sense of presence.

In discussing factors that influence our sense of presence, he also mentions concepts of efference — commands issued along the central nervous system — and afference — input from the senses. Efference and afference work together in a feedback loop which grounds us in our awareness of an environment.

He also discusses the concepts of "focal awareness" of a virtual environment, while still having "subsidiary awareness" of the real environment — using the example of being aware of the person at the other end of the telephone line during a phone call, but still being aware of the real world space that one is physically occupying. A stronger subsidiary awareness will obviously detract from a focal awareness, but the complete absence of the former is not necessary for the achievement of the latter. [31]

From this understanding of presence, Slater et al. conducted a number of investigations into a users sense of presence in virtual environments, putting their subjects through a number of experiences using virtual reality headsets to test their degree of presence. Participants were questioned after the experience, and in some cases questioned again a period of time after the experience to investigate their memories of the experience. The questionnaires developed included questions where users would rate a certain aspect of the experience on a scale, as well as more open-ended questions. [54, 55, 56]

Witmer and Singer developed a questionnaire system for evaluating presence in virtual experiences by creating two questionnaires. The ITQ (Immersive Tendencies Questionnaire) attempts to establish the immersive tendencies of study participants — such as their propensity for involving themselves in tasks, entertainment media and their own thoughts, and the degree from which they disconnect from reality while doing so. Then, to evaluate the sense of presence in experiences presented in a study, participants would answer a PQ (Presence Questionnaire) for the experience. [73] Both of these questionnaires used mostly numerical answers on a scale from 1-7 for users to select an appropriate answer between opposing extremes. The resultant answers for the PQ could then be broken down into groups to evaluate specific features of the experience, and results could be compared to the ITQ results for each participant to find correlations between certain immersive tendencies and their effect on presence in the user's perception of experiences to which they were subjected. [73]

Usoh et al. evaluated these questionnaires alongside one of their own (the Slater-Usoh-Steed questionnaire [68]), using participants assigned to various experiences to compare presence differences between experiences — desktop, virtual reality and a real world physical experience. Their results concluded that while comparisons between subjects within one type of experience could be useful, comparing the results of groups doing different experiences was less so, as users tended to normalize to their experience. Using Witmer and Singer's questionnaires, no significant differences could be found to distinguish between the experiences given to different groups, while the Slater-Usoh-Steed questionnaire only showed a small statistically significant difference, although this was mainly attributed to only 2 questions. [69]

More recently, Waltemate et al. did an investigation into the effects of avatar personalization on presence. They found that avatars that more closely resembled a users real world appearance — as opposed to a generic, equal-quality equivalent — would strongly increase a users sense of virtual body ownership, virtual presence and dominance, resulting in a stronger immersion and sense of presence. [70]

This body of work gives us an understanding which will allow us to evaluate the more subjective aspects of the final result of this system in the context of VR, and the questionnaires and methods examined here may be a useful method by which to obtain feedback from users about the subjective quality of the system created for this study.

2.5 Summary

The method of directly using a scan or some processed form of seems to be more commonly used in telepresence systems than the anthropometric body models. The latter does appear to create a scan which appears more computer generated, and therefore less realistic, and has the additional disadvantage of requiring the development or use of additional software and resources to create a system of which builds human body models in this way.

In terms of creating a model capable of being animated from direct scans, Li et al. present an interesting approach: that of obtaining a scan and using a corresponding skeleton to section the scan into multiple cage meshes which are aligned with the skeletal components which could be used to pose or animate these components. Such a form of animation could result in a much simpler transmission data for an animation across a network, requiring only data for the skeletal components. Such a method could be an efficient alternative to the approach used by Mekuria et al. of transmitting compressed fully body meshes for every frame.

The review of sensors and motion capture systems clearly supports the choice of the Xbox Kinect sensor's advantages as a device for the proposed system of this study. Systems built by Tong et al., Kreylos et al., Mekuria et al., Kuster et al. and Li et al. also demonstrated systems for direct body scans using the first generation Kinect sensor. The second generation Kinect sensor also has the advantages of increased quality over the previous generation device, and the low cost and high availability of the device make it a logical choice for systems of this nature.

Witmer and Singer's questionnaires appear to cover the concept of presence in VR very broadly, as a subjective experience based on various factors. The ITQ provided with their questions allows a profile for participants in the study to be created, which could provide insight in examining their feedback on the system. The criticism of these questionnaires by Usoh et al. in their study, claiming that the questions were a poor indicator of differences in a sense of presence when comparing different experiences across different groups of people, is noted. However, the questions suggested by Usoh et al. were not a much stronger indicator, were few in number and were fairly specific to virtual environments of their study. The questionnaires put forth by Witmer and Singer do however appear to be more informative when distinguishing between the same experience viewed by different people, or different experiences viewed by the same person, and have the added advantage of being more intuitive to answer for study participants.

Chapter 3

Background

The proposed system of this study requires a few key features:

- The ability to scan a body and create a digital 3D representation.
- The ability to transfer this scan repeatedly over a network.
- The ability to render this animation on a virtual reality display.

This purpose of this chapter is to provide background information on the concepts of virtual reality, 3D scanning, 3D animation streaming and the hardware and software chosen for construction of the proposed system.

3.1 Virtual Reality

In the broadest terms, the concept of virtual reality is an illusion of reality created by any simulated environment in which the user feels present enough in that environment that they consider it real rather than fabricated, consciously or subconsciously. The term was first used by French playwright Antonin Artuad in 1938 to describe the suspension of disbelief required from theatre audiences to be fully engaged in performances. [5]

3.1.1 History of VR

Jaron Lanier popularized the usage of the terms to refer to the virtual experiences created by digital display devices in the 1980s as founder of VPL research, a company which developed several virtual reality technologies. [48] The term also appears in science fiction, such as Derrick Broderick's The Judas Mandala [7] and Neal Stephenson's Snow Crash. [61] Various devices have been considered to present virtual reality experiences, the earliest, perhaps, being Morton Heilig's 1962 creation, Sensorama — a mechanical device which used a hood containing visual displays, binaural audio, a scent directed by a breeze and a means of inducing "vibrations or jolts" to "simulate an actual experience realistically." [14] In 1968 Ivan Sutherland created the what is widely considered to be the first virtual reality head mounted display. The portion to be worn by the user was so heavy that it had to be suspended from the ceiling, inspiring its name, the Sword of Damocles, after the Greek myth. [63] In 1991, the concept of a CAVE (Cave Automatic Virtual Environment) was developed by Cruz-Neira et al. The CAVE consisted of multiple connected projections onto the walls of a room, turning that room into a virtual environment. [10] That same year, Sega released the Sega VR headset and a company called Virtuality (formerly W Industries) also released a similar product, both intended for use in gaming arcades. Sega subsequently released the Sega VR-1 in their SegaWorld arcades in 1994. [18]

In 1995 Nintendo released their Virtual Boy console, which was widely considered to be a critical and commercial failure, due to severe limitations in the technology used and a lack of media available for the platform. [13] After 1995, the media hype surrounding VR began to fade, as the technology had seemingly failed to live up to the imagined potential. With that, public interest and potential funding dropped, resulting in very few innovations of note occurring the field of VR for quite some time.

3.1.2 Recent Virtual Reality Technology

It was not until 2012, with the official launch of Palmer Luckey's company Oculus VR, that another commercial attempt to release a VR product to the general public would be made. Funded through the crowd-funding platform Kickstarter, the Oculus Rift — originally intended to be a DIY kit for fellow VR enthusiasts — developed into a more serious commercial product after notable members of the gaming industry expressed their support, with John Carmack of id Software using the prototype to demonstrate id Software's Doom 3: BFG Edition at the Electronic Entertainment Expo in 2012. [49]

After promoting the Rift more extensively and demonstrating the device at many gaming expos, the Kickstarter campaign succeeded in raising $2.4 million (US), 874% more than the funding campaign's original target. In March 2013, the Oculus Rift Dev Kit 1 was released, followed by the Dev Kit 2 in July 2014. The first consumer model, priced at $600 (US), shipped to customers on 25 March 2016. [49] The latest headset from the company, the Oculus Rift S, was released in March 2019 and features a higher resolution than the previous headset, a special new set of lenses designed to eliminate visual errors and various ergonomic improvements to make the headset more comfortable for users. It has, however, been criticized for having a relatively low resolution compared to its competitors, and having a reduced frame rate compared to previous iterations of the Oculus Rift headset. [50]

Valve's Michael Abrash had been involved with research into VR, which Valve freely shared with Oculus in the interest of advancing VR for personal computers. [1] Valve had been working on their own VR HMD prototypes, and at some point in 2015 Valve partnered with HTC to bring a VR headset to market. This device, the HTC Vive, released in June 2016 with a price of $1200 (US), with Road to VR (a Virtual Reality News site) estimating over 100 000 units having been purchased based on Steam VR platform data. [12]

Sony had also been working on a VR device, codenamed Project Morpheus, which they released in October 2016 as the Playstation VR. [59] The introductory price was $400 (US), and as of March 2019, 4.2 million units have been sold worldwide. [53]

The HTC Vive was the first consumer VR device to receive an upgrade in the form of the Vive Pro, with improvements to the resolution, as well as various ergonomic tweaks and other hardware upgrades. [72] They also released an upgrade called the Vive Pro Eye, which features built-in eye-tracking needed for a new VR rendering method [60] and the Vive Focus, a battery powered standalone VR device. [47] A new model, the Cosmos, was briefly announced in January 2019 and is expected to have multiple improvements. [47]

3.2 3D Scanning

Various methods exist for creating a digital representation of the shape of an object in 3D space. These methods can be categorized by whether the capturing device requires contact with the surface being scanned.

Contact methods generally produce highly accurate scans, but they do have several limitations. A scanner which must make contact with an object to scan it usually requires the object to be stationary, and may interfere with the motion of a moving object. Additionally, contact scanners may deform soft surfaces while scanning them, and delicate surfaces subjected to a contact scanner may even be damaged. [24]

For the purposes of a moving object, such as the body scan intended for this project, non-contact methods are generally preferred. They do not require contact with the object they cannot interfere with the motion of an object, nor distort or damage its surface, and are able to scan objects a great deal faster than contact method devices, as they only require the movement of light, and not that of mechanical components. Non-contact scanning methods can be further broken down into active and passive methods. [6]

Active non-contact scanning methods require an emission of some sort of light onto the surface being scanned, and then detection of the light being reflected.

Two notable examples are the structured light and time-of-flight methods. Structured light scanning methods project a patterned light grid onto an object so that the surface of the object can be determined from the distortions in the pattern. Time-of-flight scanning methods, which use emissions of light pulses, measure the time which a pulse of light takes to return to the scanner, and uses that time to calculates the distance of the point on the surface of the object at which the light pulse was directed.

Passive non-contact scanning methods require multiple 2D images of the objects to be scanned. Two notable examples of this method are stereoscopic scanning and silhouette scanning. Stereoscopic scanning, which is based on similar principles by which the visual systems of humans determine depth, uses images from two cameras which are set a small distance apart to match points in the images and use their offsets to determine the distance of the points from the camera. Silhouette scanning, which requires multiple photos of a scanned object to be taken from several different angles against a high-contrast background, uses intersection of extrusions of these silhouettes to form a hull approximation of the object being scanned.

Non-contact systems have varying degrees of accuracy, and active scanning systems can have distance ranges with upper or lower bounds which, when crossed, can severely affect their accuracy. Passive scanning methods require image processing after capture to determine distances of points, which can be computationally expensive.

3.2.1 Xbox Kinect Second Generation Sensor

The Xbox Kinect is an input device originally designed to be used with the Xbox gaming console. It contains a colour camera, an infrared camera, infrared lights and a microphone. The infrared camera in combination with the infrared lights acts as a time-of flight scanning system which determines distances on the points of scanned system by measuring the return times of pulses of infrared light.

The basic data sources of the Kinect include an image feed, an audio feed, and an infrared image feed. Using these, the Kinect software provides more complex sources of data such as depth maps, speech recognition, body outlines, skeleton estimation, gesture recognition and facial reconstruction. [42]

3.3 3D Meshes

A polygon mesh is the most common method of representing three-dimensional objects in computer graphics. A mesh is primarily defined by a number of vertices, which are points in three-dimensional space which define points on the surface of a three-dimensional object. Edges connect these vertices, and a group of vertices connected by edges in a closed loop form a polygon or face. Polygons sharing edges and vertices form the surface of a mesh. In modern computer graphics, polygon meshes usually consist of triangles or quadrilateral shapes, or some combination of the two, in large numbers, to achieve detailed surfaces.

When discussing meshes, there are two main attributes that are commonly referred to: Geometry and Topology. The geometry of a mesh is the location of its vertices which gives the mesh its shape. Operations which move the vertices of a mesh, and appear to change its shape are geometric operations. Geometric operations are useful for tasks such as sculpting or animating a mesh. The topology of a mesh is the layout of its vertices along the described surface as well as the interconnection of those vertices which describe the faces. Operations which remove and reorder vertices along the described surfaces, as well as changing the layout of the faces of that surface, are topological operations. Topological operations are useful in changing the complexity of a mesh, such as compression to make the mesh contain less data by reducing the number of vertices, or adding vertices to allow a mesh to represent a finer level of detail. Images or colour values are also applied to the polygons of a mesh, in order for the mesh to display surface colours and textures of the objects which they are made to represent. [21]

3.4 Network Animation of 3D Meshes

Various methods were considered for streaming an animation of a human body scan over a network. A popular method in the research field of "3D Television" systems, is a direct streaming of multiple mesh frames, by which an animation is created when the meshes are displayed in sequence over a period of time in a manner which follows the same basic principle as video. This method is advantageous when transmitting scanned data, as unprocessed scanned data will essentially be a time varying mesh. However, the data requirements for storage and transmission of the data, without additional methods to compress or otherwise reduce the mesh data, would be fairly high.

A method which is popular in the gaming industry, although is seldom conducted over a network, is the use of dynamic meshes which are animated by a skeleton to which the mesh is connected. The movement of the mesh vertices are linked in some manner to the movement of components of an underlying skeleton, and the animation of the mesh is achieved by repositioning the components of the skeleton.

There are obstacles to using this method with meshes which are scanned rather than created, as series of meshes from a scanning device will be unlikely to have the same mesh topology. This method does, however, have the advantage of vastly lowering the data requirements for transmission and storage.

3.4.1 Direct Streaming of Multiple Mesh Frames for Animation

Direct streaming of a series of mesh frames to achieve an animation necessarily involves much larger quantities of data, when compared to similar 2D animations. Using this form of animation requires methods of reducing the amount of data which must be transmitted or stored.

For time constant meshes, where the connectivity of the series meshes remains the same but the positions of the vertices change, a popular method of achieving this is key-mesh animation. In key-mesh animation, only a few essential animation frames are transmitted, and frames of the animation between these key meshes are created by interpolation of vertex positions. [57]

For time varying meshes, where the vertices of the meshes do not correspond to vertices in meshes from other frames, key mesh animation cannot be used, so the main method of reducing data is compression of the mesh. An example of this would be the system developed by Mekuria et al. which uses an efficient method of SVA (Shared Vertex Analysis) to detect patterns in the connectivity of the mesh and codes the mesh for compression using those patterns. [40]

Both methods can deliver high quality animations and greatly reduce the amount of data required to achieve these animations. 3D scanning would likely produce a time varying mesh, so if a key mesh animation approach were to be used, another method would have to be found to create a time constant mesh from the scans. Additionally, the use of mesh compression is not limited to time varying meshes, and could be used in conjunction with key mesh animation to achieve the use even smaller amounts of data in transmission or storage of the animations.

3.4.2 Skeletal Animation of a Mesh

A common animation technique in virtual environments is to manipulate a mesh that was created statically by binding the vertices of the mesh to a skeletal structure. Bones of this skeletal structure can then be moved and kinematics can be used to determine the movement of the whole structure, which will then determine in what way the mesh deforms. Binding of the vertices to the skeleton to achieve realistic deformations of the mesh can, however, be a complex process, usually achieved through software assistance, and scanned meshes are not often animated in this fashion. [21]

Since the Kinect sensor and SDK provide an estimation of the skeleton of a body being scanned by it, the Kinect is a popular choice for markerless motion capture, and can provide a skeletal representation of reasonable quality for the purposes of animation, as pointed out in the review by Shingade et al. [52] It is possible to use the skeleton data and mesh data provided by the Kinect sensor to break down a body scan into several sub-meshes, as was done in the system developed by Li et al. [29] These sub-meshes can then be bound to the position and rotation of bones or joints, which can be obtained or calculated from the Kinect skeleton data. Then the cages can be linked and altered during the course of the animation to produce a realistically animated body scan.

3.4.3 Laplacian Mesh Deformation

Laplacian deformation was used by Li et al. to deform cage meshes and the meshes linking them to achieve a more realistic movement of the mesh surface during animation.[29] A mesh, as discussed earlier, can be described by an array of vertices V of length n, and the connections between these vertices which form the faces of the mesh. The i_{th} vertex in the list would be v_i, and the set $\mathcal{N}_i$ contains the vertices which shared an edge with v_i, and we call this set the neighbourhood, and it's elements the neighbours of v_i. The Laplacian representation of a matrix represents location of the vertices with respect to the centroid of their neighbours. The Laplacian representation of a single vertex v_i is δ_i where:

$$\delta_i = \mathcal{L}(v_i) \tag{3.1}$$

We define the Laplacian with uniform weights, which according to Sorkine is more than sufficient for meshes which are uniform. Since the Kinect detects points on a grid, the resultant mesh was entirely uniform in layout.

$$\mathcal{L}(v_i) = v_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \tag{3.2}$$

To obtain the Laplacian representation of a matrix, we wish to use the Random Walk Normalized Laplacian matrix which is:

$$L = I - D^{(-1)} \times A \tag{3.3}$$

D is referred to as the degree matrix, which is a diagonal matrix where the values of the elements in the diagonal are the number of neighbouring vertices the corresponding vertex has, which is the cardinality of $\mathcal{N}_i$.

$$D_{ii} = |\mathcal{N}_i| \tag{3.4}$$

A is referred to as the adjacency matrix, and is an $n \times n$ matrix that has a value of 1 where v_i and v_j are neighbours, and 0 otherwise.

$$A_i j = \begin{cases} 1 & \text{if } V_i = V_j \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Once we have obtained our Laplacian matrix, it can be used to give us the set of δ that represent the vertices of the mesh.

$$\Delta = LV \tag{3.6}$$

Where Δ is the Laplacian representation of the vertices in V, and each row of delta corresponds to the vertices in v.

$$\Delta[i,:] = \delta_i = \mathscr{L}(v_i) \tag{3.7}$$

To obtain a set of vertices V from this representation, we need only fix one point and solve the set of linear equations to end up with a set of vertices relevant to that one point, since the rank of L is $n-1$. To perform changes to the mesh, the approached detailed by Marc Alexa is to fix several points of the new desired mesh V', and solve for the remaining vertices by fitting the geometry of V' to that of the Laplacian representation Δ. [2]

In the transformation matrix Ti:

$$Ti = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

We wish to minimize:

$$||A_i(s_i, h_i, t_i)^T - b_i||^2 \tag{3.9}$$

Since:

$$Ai(s_i, h_i, t_i)^T = T_i V \tag{3.10}$$

A_i contains the position of v_i and its neighbours:

$$A_i = \begin{pmatrix} v_{k_x} & 0 & v_{k_z} & -v_{k_y} & 1 & 0 & 0 \\ v_{k_y} & -v_{k_z} & 0 & v_{k_x} & 0 & 1 & 0 \\ v_{k_z} & v_{k_y} & -v_{k_x} & 0 & 0 & 0 & 1 \\ \vdots & & & & & & \end{pmatrix} \tag{3.11}$$

and b_i contains the position of v'_i and its neighbours:

$$b_i \left(v'_{k_x} v'_{k_y} v'_{k_z} \vdots \right) \tag{3.12}$$

The linear least-squares problem above is solved by the equation:

$$(s_i, h_i, t_i)^T = (A_i^T A_i)^{-1} A_i^T b_i, \tag{3.13}$$

Since we know the values of A_i from the initial mesh V, s_i, h_i and t_i are therefore linear functions of b_i, which allows us to obtain our transformation matrix T_i from setting one point (or more) to a new location and solving the set of linear equations to obtain the transformation that must be applied to other points in V to obtain the new set of points V'.

3.5 Unity Game Engine

Unity is a game creation engine that provides various tools for the creation and display of virtual objects and environments. It has a physics engine and provides the ability to render 2D dimensional textures and 3D objects. Positioning is controlled by Unity vector objects in 2D and 3D space: i.e. the Vector2 and Vector3 objects, which are composed of double values. The Unity game engine, among other functionality, provides a framework for the creation of virtual environments. At the time of construction of the system for this study, Unity had already integrated virtual reality support for the Oculus rift into the game engine, and the development team of the Kinect SDK had created a Unity project template for use with their system. These two factors strongly suggested the use of Unity over other alternatives, which will be further discussed in Chapter 4.

3.5.1 Meshes in Unity

Unity meshes are useful in representing 3D objects, and they contain the following components:

- Vertices: Vertices are described by a vector in three-dimensional space, represented by Unity's Vector3 data type, and the geometry of a Unity mesh is defined by an array of these vertices.
- Triangles: The topology of Unity meshes is defined by a series of triangle faces that connect 3 vertices. This data is stored in an array of integers, where each set of three consecutive integer values is the index value of the vertices in the vertices array from which the triangle is created. The array cannot have a higher value than the length of the vertices array.
- UVs: UVs describe, by means of a double between 0.0 and 1,0, at which point on a two-dimensional colour image a certain vector takes its colour from. This is represented by an array of Unity's Vector2 data type, which describes a vertical and horizontal location on the texture image. The array

is the same length as the vertices array, and each UV corresponds to the vertex at the same index value.

- Normals: Normals are the normal direction of the faces of the vertex, and determine in rendering which side of the face of the mesh will be visible. This is stored in an array of Vector3s which correspond to the triangles specified in the triangles array. Unity provides a function to recalculate this for the mesh. The array is one third the size of the triangles array, and each normal corresponds to a set of 3 index values that represent a triangle.

Unity's mesh object allows the creation of a mesh object with only vertex and triangle data. UV data is required for the application of texture and normals can be supplied or calculated by a function in the mesh class.[65]

3.5.2 Kinect in Unity

Microsoft had previously released the first generation of the Kinect sensor, in an edition specifically for development on Microsoft Windows devices, as well as an adaptor for the device, as it was originally intended for the Xbox 360 gaming console. This was accompanied by the Kinect SDK (Software Development Kit), a software library which allows for the gathering of several data sources, both unprocessed inputs, and more complex sources which are generated from the raw data captured by the Kinect, sometimes building on each other. The second Kinect, designed for the follow-up console, the Xbox One, was also released with an adaptor for Windows computers and the Kinect SDK2 development software. Windows and Unity subsequently released a project template in Unity which supported the Kinect V2 sensor, enabling the Kinect Sensor to run on the Unity platform as it would in normal .NET applications. [42]

3.5.3 Virtual Reality in Unity

Unity has built in VR functionality, which supports the Oculus Rift, as well as other VR devices. It can be enabled through project settings in the Unity Editor. No additional editing of a Unity environment is required to add VR functionality to a game — Unity's main camera is automatically adapted (When VR is enabled and a device is available) to a stereoscopic feed, and the camera orientation and position are controlled by input from the VR device. [66]

3.6 Summary

This chapter provided context for the concepts of Virtual Reality, 3D scanning and 3D streaming, which feature strongly in this project. Client-side skeletal animation was discussed as an alternative to direct streaming of 3D models, as well as the Laplacian mesh deformation which could augment a cage mesh method of skeletal animation to provide a more realistic information.

Additional information was also provided behind specific hardware and software relevant to the construction of this system, such as the Unity game engine and the Xbox Kinect second generation sensor. A review of current literature, as discussed in Chapter 2, strongly supported the use of this sensor for the 3D capturing process. Further motivations for these choices and others are discussed in Chapter 4.

Chapter 4

System Design

This chapter details the selection of the hardware, software and methods for the creation of the system for this study. These choices are based on the requirements and goals stated in Chapter 1, as well as the available resources and other realistic factors.

A brief overview is also given of the intended architecture of the system, influenced by these choices and the various operations which are necessary to achieve the stated aims of this study.

4.1 System Hardware

As the basic purpose of the system is to obtain and transmit a body scan over a network to be viewed in a virtual reality environment, the two most basic hardware requirements of this system are the device which obtains the body scan and the device which displays the system in virtual reality. Virtual reality additionally comes with a need for the users to be able to navigate their environment, so a method of user control is also needed.

These forms of hardware generally do not function independently, and as such must be powered by computers. As a minimum requirement, a computer that meets the processing requirements of the Kinect second generation sensor is needed, and another computer that meets the graphical requirements of the virtual reality device chosen. Both these computers are needed in order to create a demonstration of the system transmitting a body scan over a network.

When selecting system hardware, devices that were more cost effective and more easily commercially available were preferred. Devices already available to the researcher from a previous project which met the needs of the system are used where possible.

4.1.1 Scanning Device

The Kinect second generation sensor was chosen as the scanning device for the proposed system. The review by Shingade et al. detailed many motion capture systems using the Kinect first generation sensor, and the device comparison in that review which discussed various depth and RGB cameras used for similar purposes showed substantial advantages to using the Kinect first generation sensor.[52] The Kinect second generation sensor has several improvements over its predecessor, and the SDK for the second generation sensor is also more advanced. [42] Additionally, the Kinect second generation sensor and the adaptor necessary for development on a Windows computer were widely available and reasonably priced when this study commenced, making it a fairly logical choice.

4.1.2 Virtual Reality Display Device

For a virtual reality display device, it was decided that the Oculus Rift DK2 headset would be used. At the time of the start of the project, the Oculus Rift DK2 was the most sophisticated VR device commercially available, and one was already available to the researcher, having been used in an earlier project in which the researcher participated. [32]

4.1.3 User Control

For user control, a system from a previous project was considered, using the Leap Motion controller mounted to the front of an Oculus Rift. [32] However, in that project, the Leap motion controller provided interactive controls, and as a hand-motion controlled device made less sense as a device for providing navigational controls, which this project would require. A simpler control method — a windows compatible game controller, the Logitech F710, was used instead.

4.1.4 Development and Display Computers

An Intel NUC computer was chosen as the device which would power the virtual reality display, as the small form factor made it ideal for fast set up of VR demos, and a similar device had performed adequately an earlier project in which the researcher participated [32], proving able to run the Oculus Rift DK2 headset without serious performance issues or graphical errors. The NUC for the project had an Intel i7 processor, 8 GB RAM and Intel integrated graphics. An additional computer was required to process the scan data and send it across the network. A Lenovo Y5070 laptop was used, with an Intel i7 processor, 4 GB RAM and an Nvidia GTX 860M graphics card.

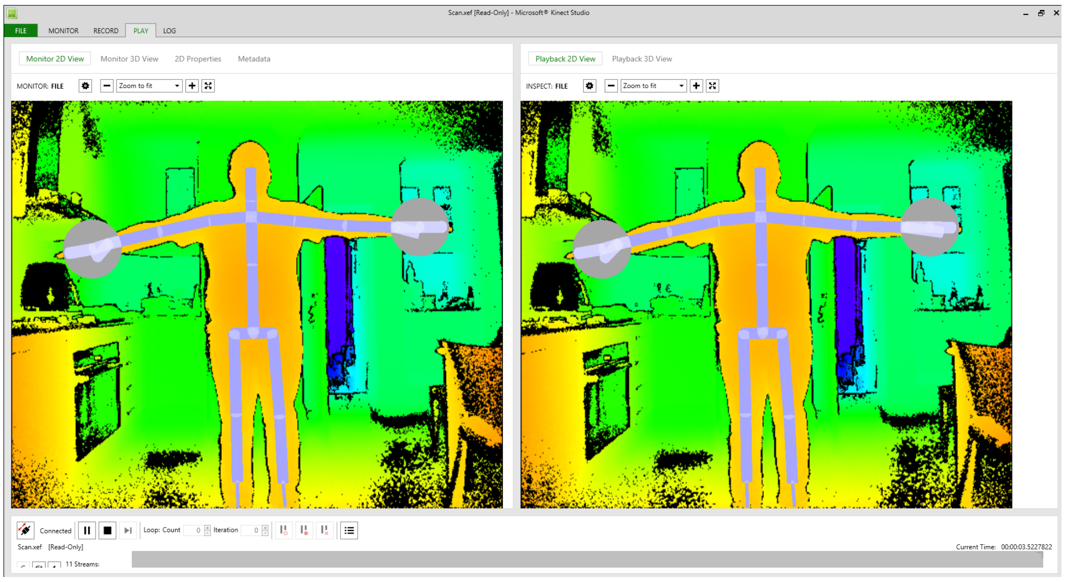


Figure 4.1: Kinect Studio Tool.

4.2 System Software

Software for interaction with the selected hardware was required, particularly software for receiving data from the Kinect sensor and software for displaying the system on a virtual reality head mounted display. Additionally, a framework for a virtual environment would be needed, with commercial game engines being considered, mostly due to the strong support of virtual reality technology in gaming. A strong emphasis on software components with better potential for integration influenced the eventual choice of software.

4.2.1 Scanning Software

The Kinect SDK2 was the only available software choice for use with the Kinect second generation sensor when development of this system commenced. It supports development in all .NET languages. C# and C++ were considered for development, as these were the development languages for the game engines considered for use, which will be discussed in Section 4.2.3. The Kinect SDK also provides a utility called Kinect Studio, which allows for the recording and playback of Kinect data streams. Playback can be displayed as a 2D or 3D visualization based on which data streams are present, but this utility can most usefully also emulate a Kinect sensor using pre-recorded data. This utility therefore allowed for system development to occur without the need to always have a Kinect sensor present. This also removed the need to store Kinect data for development, as Kinect Studio can store and retrieve Kinect sensor recordings from XEF (eXtended Event File) files.

Recordings in Kinect Studio also allow for the disabling of certain sources. It is important to note, when using this utility, that certain sources from the Kinect sensor are dependent on other sources, and that the disabling of certain sources will also automatically disable any sources which are dependent on that source.

4.2.2 Virtual Reality Support

The Oculus Rift SDK requires software written in C++ to provide virtual reality support. Version 0.6 of the SDK was used, and development continued with this version throughout the project, as later versions discontinued support for powering the display device with integrated graphics, which was necessary for the chosen demonstration computer. The Oculus Rift DK2 and it's accompanying Software development kit meet the requirements stated in Section 1.5.5, making this combination an overall satisfactory choice for use in our system.

4.2.3 Software for Virtual Environment Development

Game engines were considered for the use in development of virtual environments which users could navigate, as they have built in functionality for graphical rendering, user input, physics and lighting, and various other features useful for building virtual environments. The Unity engine and the Unreal Engine 4 were both considered. The Unity game engine is programmable using Javascript or the Microsoft C# language. Microsoft have provided an asset package, as mentioned in Section 3.5.2, for the Unity engine, which allows for the connection of the Kinect second generation sensor and the ability to access its data from within the Unity Engine. The Unity engine also has built-in Virtual Reality functionality, which supports the Oculus Rift platform, as mentioned in Section 3.5.3.

Similar to Unity, Unreal engine is also programmable, using the C++ language. However, when the development of this system commenced, the Unreal engine provided no VR support. Functionality for the Xbox Kinect sensor and the Oculus Rift VR headset would have had to have been built for the system in C++.

Unity was chosen, as both the Kinect sensor and Oculus VR headset were supported by this engine, meaning no extra development would be needed to support both devices. This necessitated the use of the Microsoft C# language for development. Version 5.12 of the Unity Editor was used, as this was the version that supported version 0.6 of the Oculus Rift SDK.

4.3 Methods and Processes

Based upon the studies reviewed in Chapter 2, a few general methods were decided upon for the capture of the scan, the reduction of the bandwidth required for transmission and the manner in which the mesh would be animated across a network. Prototyping in the Unity game engine was conducted to further determine certain details of these approaches, and this influenced the eventual selection of the methods used.

4.3.1 Body Scan Method

An anthropometric approach as mentioned in Section 2.1 was ruled out, as it would require extra resources and possibly result in a scan which appeared to be generated rather than recorded. Instead, a body scan would be constructed directly from Kinect sensor data. A choice was made to use a single Kinect sensor, for a number of reasons, but foremost among these reasons was that the use of multiple Kinect sensors would not meet the goals of this study for a low-cost and simple hardware set-up.

The Kinect SDK does not allow for multiple Kinect sensors to be connected to a single computer, and as such multiple Kinect sensors would require the use of multiple processing computers. Networking infrastructure to combine these sources would have had to have been created, and additional development would be required to synchronize the frames of these systems and combine these sources.

Prototyping with the Kinect sensor showed that a fairly complete static mesh representing a human body could be obtained from two single frames of Kinect data: obtained from a user showing suitable poses of the front and back of the body while being scanned. This achieved the goals of a low-cost and simple hardware set-up, while still providing a mesh that would be complete and capable of being animated.

4.3.2 Method of Scan Transmission and Animation

Direct transmission of time varying meshes would not be possible with the use of a single Kinect sensor, as this approach would create an incomplete mesh when using only one Kinect sensor. Viewed in a navigable virtual environment, where such a scan could be viewed from any angle, it would likely break the immersion of viewers to see a body scan which was missing its back half. Ultimately, the method of animation decided upon was skeletal animation of linked “cage meshes”, similar to the approach used by Li et al., where a mesh would be split up into sub-meshes that have their movement controlled by the joints or bones which lie beneath them. [29]

4.3.2.1 Laplacian Deformation Approach

The approach of using Laplacian deformation to create a realistic movement of these sub-meshes was investigated. A prototype was implemented in the Unity engine, using a simple cylinder deformed by holding two joints in place, and moving a third, central joint. Figure 4.2 shows the prototyping of the Laplacian deformation program. Figure 4.2a shows the original mesh, Figure 4.2b shows the mesh

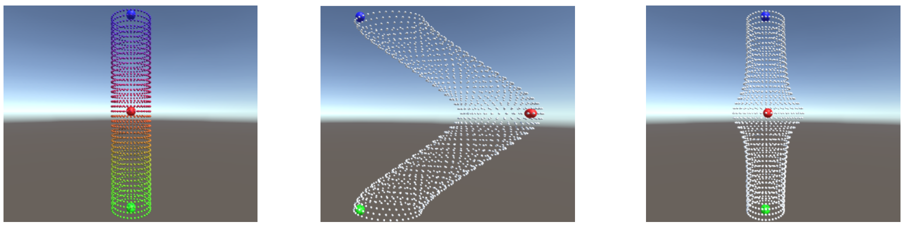

(a) The original mesh. (b) Mesh with laplacian deformation. (c) Mesh with a "reversed" laplacian deformation.

Figure 4.2: Visualisation of Laplacian Deformation Prototyping.

with a Laplacian deformation applied to follow the displaced central joint, and Figure 4.2c shows the Laplacian deformation applied again when the central joint is moved back to its original position.

The result achieved a smooth deformation, although moving the joint back to its original position did not restore the mesh to its original condition, as can be seen in Figure 4.2c.

A more significant issue, however, was that this deformation of a simple mesh with a number of vertices that was few in comparison to the body scan took over 3 minutes to execute. In an attempt to avoid any possible inefficiencies in the Unity engine, a library which efficiently implemented matrix operations was used for further prototyping of the same concept. This delivered similar processing times.

Unfortunately, this performance overhead prevented the use of the Laplacian deformation for the purposes of animation in this project.

4.3.2.2 Linked Meshes Approach

Being unable to achieve the required processing time for Laplacian mesh deformations achieved in the study conducted by Li et al., a simpler method of linking the cage meshes during animation was used. [29] The mesh would be broken down into several cage meshes, and these would be kept as non-deforming objects which were attached to skeleton bones or joints, and moved with these objects. To create the impression of a complete mesh, additional meshes would be generated to link these cage meshes, updating and deforming to match the new locations of the cage meshes.

4.4 System Architecture

As discussed in Section 1.4, the system consists of four main components:

1. Capture of the Kinect mesh and skeleton data.
2. Processing the mesh data into a body scan which is capable of being animated.
3. Transmitting the Kinect data. Once in the case of the body scan, and continuous streaming in the case of the skeleton data.
4. Displaying the body scan and animating it with the stream of skeleton data.

The modular nature of the Unity Game Engine encouraged a system design composed of many small, interoperable components. Geometric operations were vastly simplified by the interaction of scanned components in the Unity engines 3D space, which also allowed visual debugging during the development process. After processing, a system architecture was decided upon which had grouped scanning functionality, grouped networking functionality and separate pipelines for mesh and skeleton data, both in the scanning and networking section of the software.

Due to the visual nature of the debugging, and the requirements for a display for demonstrations of the system, a component that creates displays of the data and links to the Unity environment is used as both the interface for recording a scan and viewing a scan transmitted over the network. A diagram of the proposed system can be seen in Figure 4.3.

Unity's publishing functionality was used to create standalone executable software in versions which could be set to capture a scan and transmit it over a network, or to receive that scan and display it on a virtual reality head mounted display.

4.4.1 Capturing

The various data streams relevant to the body scan must be captured from the Kinect sensor using the SDK and the package provided for the Unity game engine, by Microsoft, for this purpose. The skeleton data requires little processing to obtain an appropriate form for this system. This data will be used for both the processing and the animation of the mesh.

To create the body scan the Kinect data must first be processed into Unity meshes, which requires the removal of the backgrounds of the scans, the creation of a mesh from depth data, the creation of a texture from the color data, and the application of that texture to the created mesh. This process is discussed in more detail in Chapter 5.

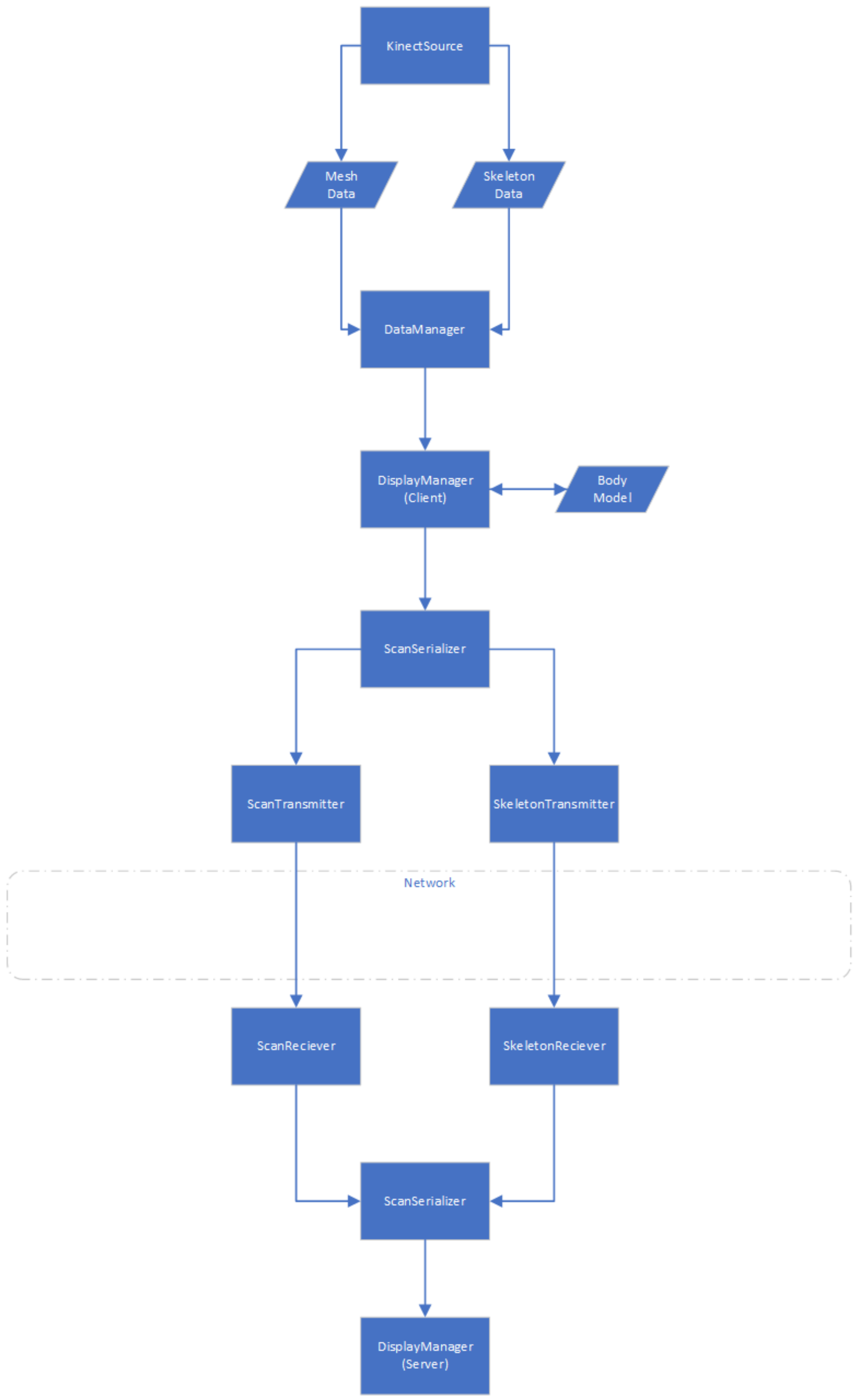


Figure 4.3: System Architecture Diagram.

4.4.2 Processing

Once frames are selected for the construction of the body model, the scans must be analysed and broken down into moveable cage meshes, which are obtained from front and back scans that must be aligned, split and recombined. Linking meshes must then be created between these cage meshes, with reference points created to control their deformation during the animation. The steps involved in this process are further explained in Chapter 6.

4.4.3 Transmitting

The processed body scan must be serialized and sent over a network to initiate the transmission, and thereafter the skeleton data must be continuously streamed to animate the scan. This is further discussed in Chapter 7.

4.4.4 Display and Animation

Once transmitted across the network, the body scan must be recreated for display and animation by an incoming stream of skeleton data frames. This is discussed in more detail in Chapter 7.

4.5 Summary

For the system, it was decided that an Xbox Kinect second generation sensor would be used as a scanning device, an Oculus Rift DK2 would be used as the virtual reality display, a game controller would be used for navigation, and computers sufficiently powerful to operate the Kinect and the Oculus Rift would be used for development and demonstration of the system. Software to be used would include the standard SDKs for both the Kinect and the Oculus Rift, and the framework for development would be the Unity Game Engine. Development would take place using the C# language.

The system would use the method of obtaining a point cloud body scan and creating a mesh from that scan, and using skeleton data to animate a processed form of that mesh over a network. The system would be modular, with groups for scanning and network transmission, and a system for displaying and interacting with body scans and the virtual environment. A system architecture diagram is presented in Figure 4.3.

CHAPTER 5

OBTAINING A BODY SCAN MESH

This chapter details the processes and methods used in the system components that are responsible for the capture of data from the scanning device, and the creation of a mesh which represents the scanned human body.

We discuss the methods by which the scan is captured, the types of data which are used in the construction of a mesh, the preparation that data requires, and the method by which a mesh representative of a human body is created from this data. We also discuss the capture and use of the correlating skeleton data which is used both for mesh processing (Chapter 6) and animation of the transmitted scan. (Chapter 7)

5.1 Capturing

The initial step in acquiring the human body scan for the system was to design a physical area in which the scanning would take place and the methods for acquiring the scan. The animation would take place in a similar area.

5.1.1 Hardware Set-up

A simple hardware arrangement was a proposed goal of the system. As such, scanning was conducted with a single Kinect second generation sensor mounted on a normal tripod set to hold the Kinect sensor approximately 1.5 meters off of the ground. The user would then stand and conduct the scanning or motion capture process standing approximately 2 meters away from the Kinect sensor.

The system allowed for a fair deal of deviation from this set-up — as long as a human body is detected by the Kinect sensor, the system will attempt to create a body scan from what it is able to record. However, for a workable human body mesh to be scanned the user must be fully in the Kinect sensor's field of view, and have no parts of their body occluded by objects or surfaces between the user and the Kinect sensor.

The Kinect itself also has a maximum depth sensing range of 8 metres, but body tracking (which provides the skeleton data necessary for mesh processing and animation) becomes unreliable after 4.5 meters. Around this limit and beyond it, the system will be unlikely to deliver a scan which resembles a human body, at least in terms of outline and joint data.

Greater surface scan quality is also achieved when the user is closer to the sensor, for which the distance of approximately 2 meters appeared to be ideal, as this allows the user to get as close as possible to the sensor while still having their entire body in the sensor's field of view.

For the body scan and animations recorded for the evaluation of this system, an additional optional measure was taken to ensure optimal quality — the recording was conducted against a flat background surface of a light colour with no objects around the user. This step, while not compulsory, removed the risk of a number of possible scanner errors.

5.1.2 Scanning Process

It was determined that a front and back scan would be sufficient for the purposes of this system, so the process created for the scanning of the body required a recording in which a mesh was obtained for both the front and back of the human body being scanned.

A pose was chosen which prevents the arms of a user from occluding parts of the rest of the body by being located in front of the user. A pose with the arms raised to the side of the user, pointing outwards was chosen. This pose can be seen in Figure 5.1. The user must first directly face the Kinect and perform this pose for a brief period of time, and then turn to face directly away from the camera and perform the same pose again for another brief period of time.

The user may wish to give another user at the computer time to select each scan if the selection is being done directly from the Kinect sensor, or if the user plans on using a Kinect Studio recording they may wish to remain in the pose only long enough to ensure that they have been in the correct pose, as the Kinect sensor captures 30 frames per second and the Kinect Studio utility allows users to step through the frames of a recording. A user observing the scanning process manually selects a suitable single frame which shows the front of the human body being scanned and a single frame for the back. This can either be done through directly using the Kinect camera or by way of playing back an earlier recording using the Kinect Studio utility. For the purposes of this study and the demonstrations conducted in the evaluation of the system, a recording was used. Figure 5.1

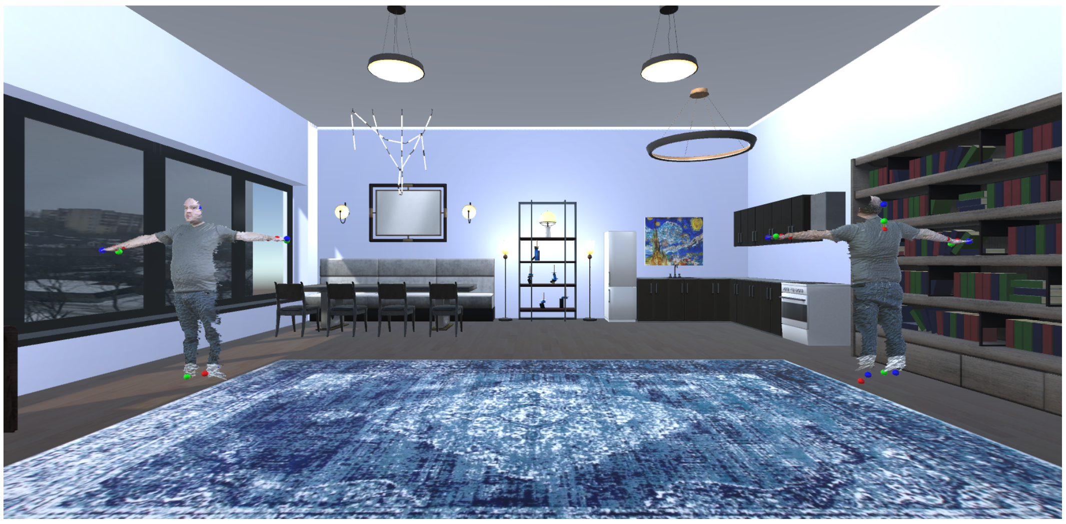


Figure 5.1: Front scan (left) and back scan (right) selected by the user, as seen in the demonstration environment.

shows the display where the front and back are scan selected by the user to create a representative body scan.

The system allows for the replacement of scans, if those selected are deemed by the user to be of insufficient quality.

5.1.3 Motion Capture for Animation

The process for capturing the data from the Kinect required to animate our body scan across the network is designed around the skeleton data source that is used to power that animation. As such, considerations which may influence the mesh quality are irrelevant in terms of the animation — as our only concern is that reliable joint data is received from the sensor.

It remains important that a users entire body remains within the Kinect sensor's field of view, although with a degree of allowance, as the Kinect will create estimates for joints which it cannot properly detect. As a result, edge joints (such as those of the hands and ankles) which are out of the field of view may still seem to be in reasonable positions and will not create a distorted animation. Users must still stay within 4.5 meters of the sensor, as joint data becomes unreliable after this point. Occlusion by objects and surfaces remains an issue, although the Kinect sensor's skeleton estimation is fairly resilient to most occlusions that do not solidly block multiple joints from the field of view.

5.2 Unprocessed Data

The Kinect sensor used in this system provides a variety of processed and unprocessed data streams which are either obtained from the sensors hardware or generated by the Kinect software from these unprocessed streams. For the creation of a mesh which represents the human bodies that our system scans, and the processing and animation of that mesh, the created system uses the following four streams.

Color The RGBA camera data, which provides the data which is used to create a texture for the created mesh.

Depth The processed infrared camera data.

BodyIndex A masking image indicating pixels where detected bodies are present.

Body The calculated detected bodies data, from which we extract skeleton information for the body scan and animation.

We use these sources in the system for the creation of the body scan meshes, the processing of those meshes and the animation of the resultant mesh. A brief discussion of the nature of each data source follows.

5.2.1 Color Data

The colour data stream from the Kinect sensor is given as a byte array that has a length of 8294400 bytes. This represents an image with an HD resolution (1920×1080) where each pixel is represented by four bytes. Each four consecutive bytes in the array represent a red, green, blue and alpha value. A visualization of this data can be seen in Figure 5.2.

5.2.2 Depth Data

The depth data stream from the Kinect sensor is given as an array of 16 bit unsigned integers representing distances, in millimetres, from the camera. The resolution of this depth image is 512 by 424, resulting in an array of 217088 integers. A visualization of this data can be seen in Figure 5.3.

5.2.3 Body Index Data

The body index data stream is a masking map which indicates which pixels in the depth frame correspond to a tracked body, and which pixels correspond to the background. This data is generated from the depth image, so has the same resolution of 512 by 424, and is an array of 8 bit unsigned integers, where values

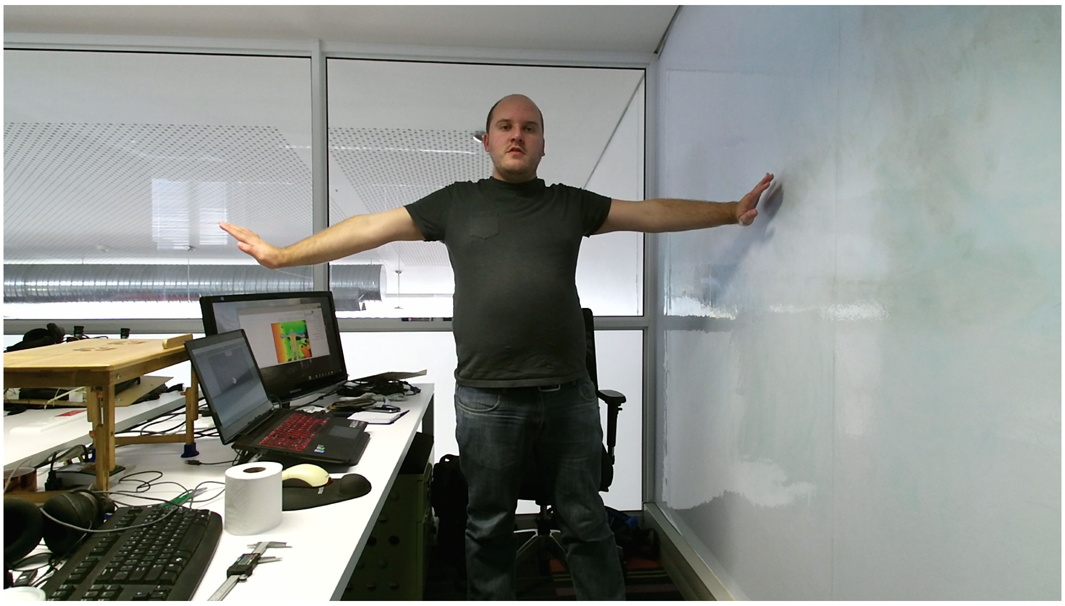


Figure 5.2: Kinect SDK Visualisation of a Kinect Color Data Frame.

of `0`-`5` map to tracked bodies, and greater values represent background pixels. The system is designed to track only the first body tracked by the Kinect sensor, so values of `0` were taken as an indicator of pixel representing a body, and all other values were considered to be background pixels. A visualization of this data, for a single tracked body, can be seen in Figure 5.4.

5.2.4 Skeleton Data

The skeleton data for the purposes of processing and animation is obtained from a Kinect source containing an array of Body information. The Kinect sensor can track up to six human bodies, and provides various information about each tracked body.

The primary information includes joint positions, joint rotations, hand gestures, clipped edges (where joints in a tracked body leave the Kinect sensor's field of view) and indicators of the direction in which a body is leaning. As stated in Section 5.2.3, this system was designed to transmit a scan of only one human body. Only data from the first detected body in the frame is used, and subsequent bodies are discarded. A visualization of this data, for a single tracked body, can be seen in Figure 5.5. For the selected body, joint positions and the joint rotations were stored, although only joint rotations for the spinal joints proved relevant to the system.

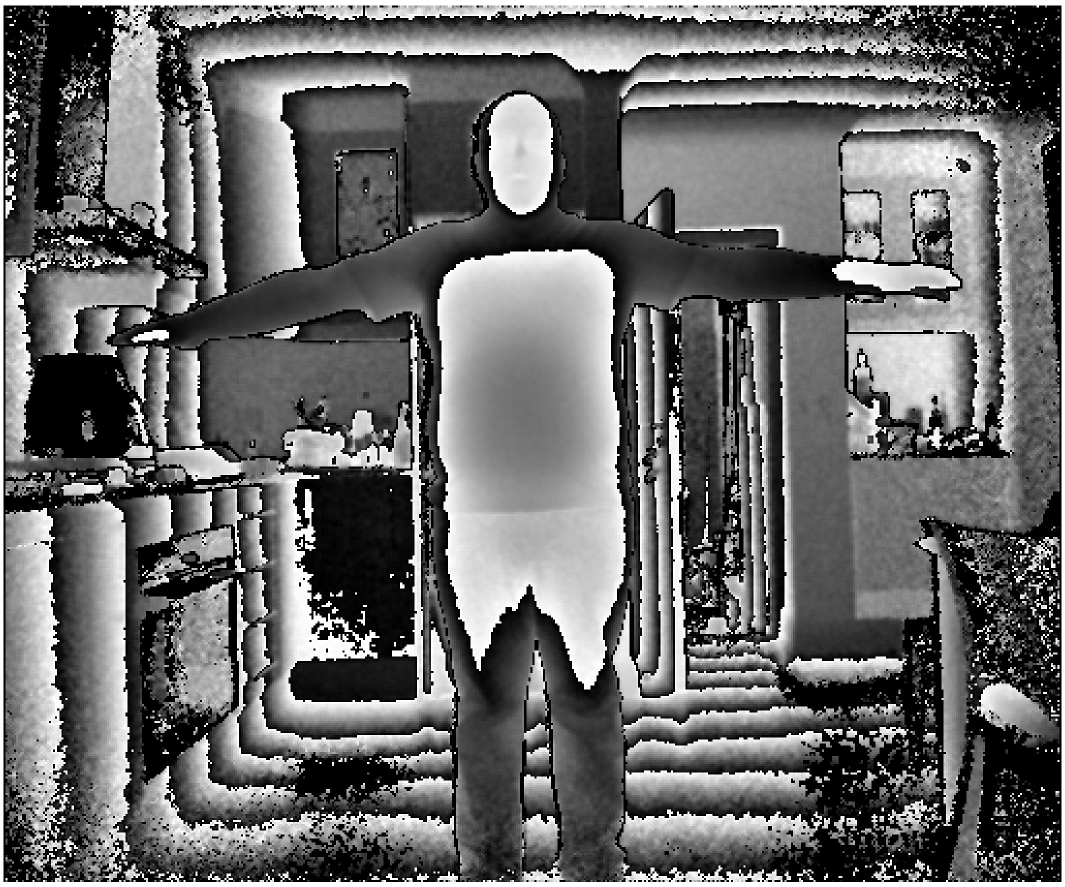


Figure 5.3: Kinect SDK Visualisation of a Kinect Depth Data Frame.

5.3 Preparing Mesh Data

Once the data is obtained from the Kinect sensor, it must be prepared for use in the mesh creation process which will take place in the Unity environment. The primary concerns are the removal of background data, and the removal of any noisy or erroneous data which would complicate the creation of the mesh.

5.3.1 Eliminating Outliers

Before any of the depth or color data is used, the body index data is cleaned to remove noisy data or objects near the scanned user that may have been erroneously registered as part of the scanned body.

The noisy data generally takes the form of small isolated spots appearing across the image, and is usually a result of poor lighting conditions. Objects erroneously identified as part of the scanned body are usually objects very close next to a scanned body but not close enough to be connected to it by pixels. However, objects between the user and the Kinect sensor will occlude parts of the scanned body, a problem for which this system does not provide a solution.

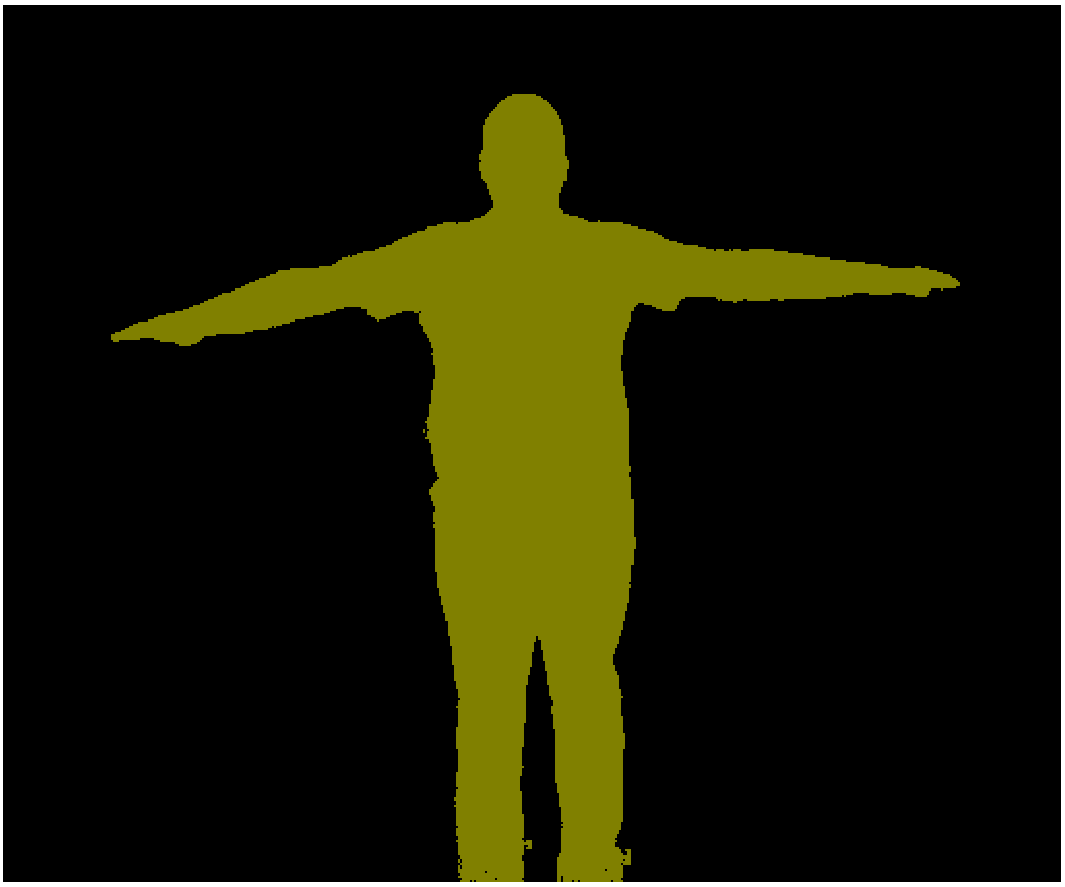


Figure 5.4: Kinect SDK Visualisation of a Kinect Body Index Data Frame.

Elimination of these outlying points or groups of points is achieved by a simple method of alternate erosion and dilation of the masking image obtained from the Body Index data. Both methods loop through the body index array to count, for each member of the array, the number the surrounding members which are empty — those pixels that do not represent a body — and those which are non-empty — and do represent a pixel on the body. Erosion removes points on the body index array if they were bordered by a sufficient number empty pixels. For the purposes of this system, four pixels were required. Dilation restores points on the body index array if they are bordered by a sufficient number of non-empty pixels. For the purposes of this system, three pixels were required. The pixels are only restored if they were originally present. This can be done any number of times, and once all iterations of erosion and dilation are completed, the maximum region — the largest number of connected pixels — is identified, and any other regions (should they exist) were removed.

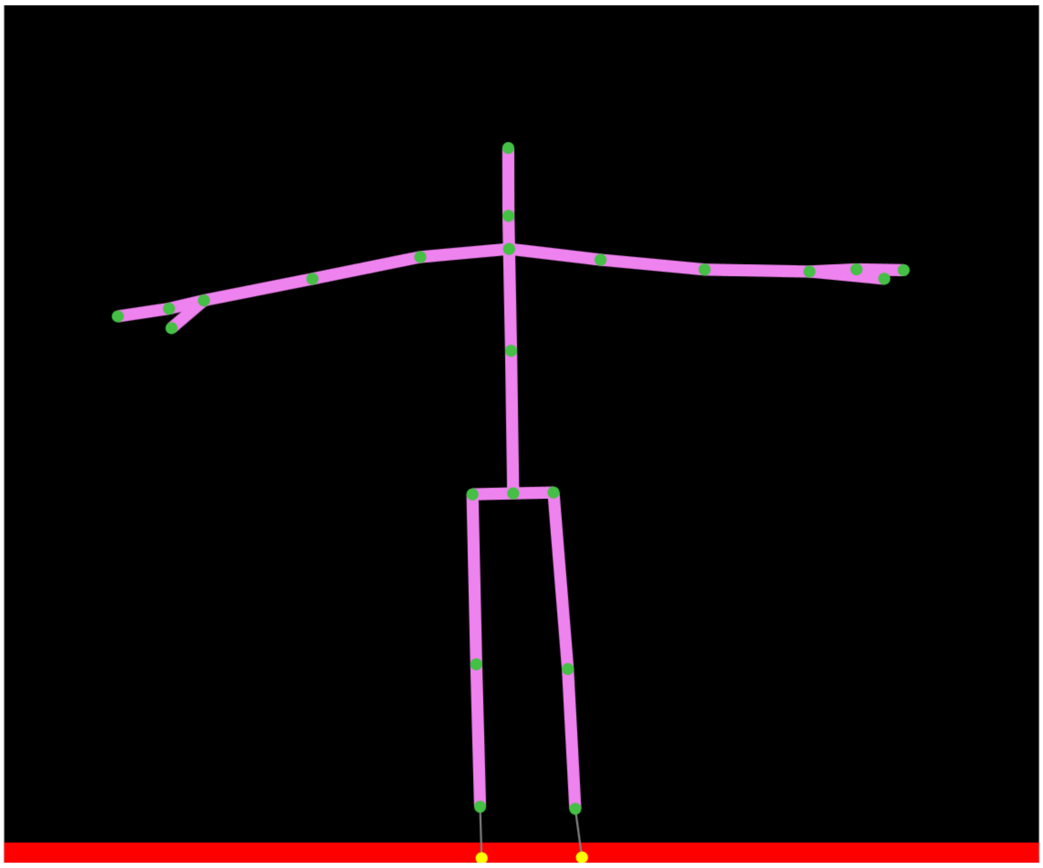


Figure 5.5: Kinect SDK Visualisation of a Kinect Body Object.

5.3.2 Background Removal

The infrared and RGB cameras of the Kinect sensor are located a small (but not insignificant) distance apart from each other on the physical device, and the depth data does not follow the exact coordinates of the infrared source of the sensor, while the body index source does. This results in the three sources having different coordinate systems which are not aligned or necessarily proportional to each other. To use these three feeds in conjunction to eliminate background depth and color data, it is necessary to be able to compare each pixel. Fortunately, the Kinect sensor provides functionality for such comparisons in the form of coordinate mappers — which allow an index of a pixel to be used to find the corresponding pixel on another. Every pixel of the depth data and the color data are mapped onto their corresponding pixel in the body index data, after the outliers have been removed from this data by the process discussed in Section 5.3.1. Mappings which indicate that a pixel of depth or color data do not represent a scanned body are identified, and these are used to remove both sets of background data for both sources.

5.4 Mesh Creation

Once the data has been prepared — removing the background with a mask which is cleared of outlying or noisy data — the body scan mesh can be generated.

Using the depth data we create those components which define the geometry of the mesh and create its 3D surface — the Vertices and the Triangles. Thereafter, the color data is used to create a texture which will cover the mesh and allow it to render the mesh in color, which allows us to fully realize the body mesh in the extent of detail that the scanner can provide.

5.4.1 Creating the Mesh Vertices

With the background removed, the depth data gives us a point cloud representation of the human body that has been scanned, where each member of the array is a point in 3D space on the surface of the scanned body. These points are used to create the vertices of the mesh in Unity.

5.4.2 Creating the Mesh Triangles

Once a set of vertices for the mesh is created, triangles for the mesh surface can be generated. Fortunately, as the vertices are created from uniform grid of depth values, they provide a uniform matrix. With a point cloud of vertices which is uniform, it is a simple matter to generate triangles using the marching squares method.

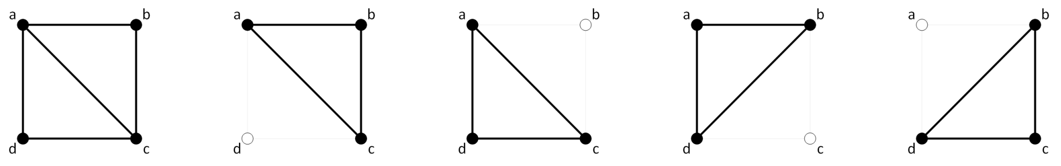
5.4.2.1 Marching Squares Method

The marching squares method moves through a uniform array of vertices, examining each group of four adjacent elements of the array to determine whether those elements contain a mesh vertex or are empty. Should they contain a mesh vertex, that point can be used to form a triangle. The pattern formed by the four vertices examined determines the number of triangles created and which vertices they connect to.

Figure 5.6 shows the only five patterns of empty and non-empty vertices where triangles are created. Figure 5.6a, shows that when all four vertices are non-empty, two triangles are created. Figures 5.6b, 5.6c, 5.6d and 5.6e show that when only three vertices are non-empty, only one triangle is created. There are eleven other possible patterns, which do not create triangles, as they do not have a sufficient number of points.

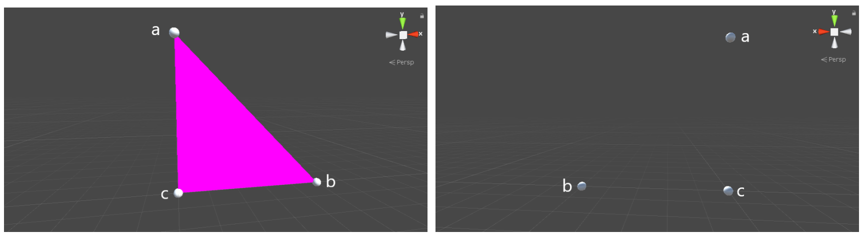
5.4.2.2 Mesh Normals and Triangle Order

Unity stores meshes as an array of integer values that are the indices of members of the mesh vertices array. Each consecutive three value describes a triangle by



(a) (b) (c) (d) (e)

Figure 5.6: Marching Squares Patterns where Triangle Creation is Possible.



(a) The face from which the order of the vertices is clockwise. (b) The face from which the order of the vertices is counter-clockwise.

Figure 5.7: View of a Generated Triangle in Unity from Opposite Angles.

referencing the three vertices it connects. Unity renders mesh triangles in a single direction based on the order in which the points of the triangle are specified. This means that the triangle will be visible from one side and invisible from the other. The visible face of the triangle will be from the direction in which vertices are arrayed in a clockwise direction in the order in which they were specified. Figure 5.7 shows the triangle created in the order a-b-c, from the clockwise and counter clockwise faces.

5.4.3 Applying a Texture to the Mesh

A texture was created for the mesh from the color stream of the Kinect sensor's data. In order to apply this texture to the mesh we have created, we must create an array for the mesh UVs — coordinates for the mapping of a texture to the mesh. UVs are stored in array of matching length to the array of vertices, and each UV directly corresponds to the vertex with the same index. Each UV describes the horizontal and vertical position (by percentage) that should be aligned to the corresponding vertex. With a full set of values, the texture is then stretched over the surface of the mesh, using the UVs at the vertices to manage the way in which the image is adapted for the 3D surface. To create this mapping, we use the same coordinate mapping functionality of the Kinect software as discussed in Section 5.3.2. We map the depth image to the color image, and use that mapping to calculate the UVs, which we then store in the UV array of the mesh. A representation of the resultant mesh object is displayed in Figure 5.8.

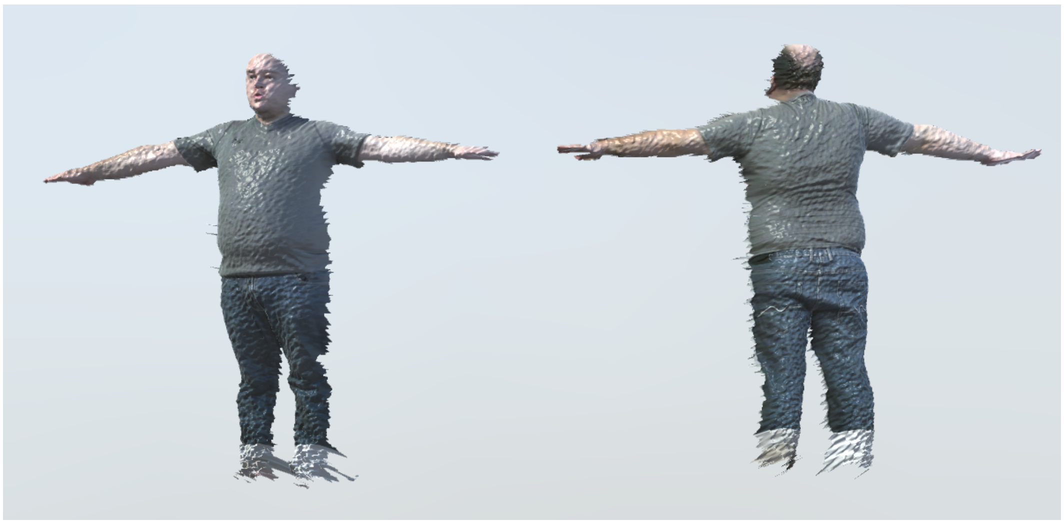


Figure 5.8: Visualisation of Unprocessed Front and Back Mesh Data.

5.4.4 Creating a Representation of the Skeleton

For the purposes of our system, skeleton data requires less manipulation than the data necessary for the mesh, as from the Kinect source we can obtain both the points in 3D space where the sensor estimates the location of joints, and the estimated rotation of those joints. However, some selection is done to come to this data set — the Kinect sensor is capable of tracking multiple bodies, and this system only attempts to record one, so the first detected body is selected. Additionally, representations of bones were created, at midpoints between joints, to be extra points of animation for the body scan, and provided a finer degree of detail for the analysis of the mesh. Both joints and bones were used in this analysis of the mesh during processing to identify which regions of mesh would attach to which moving points, which will be further discussed in Chapter 6. They will also both be necessary for the animation process, which will be further discussed in Chapter 7.

5.4.4.1 Joints

The joint objects used in the system are primarily important as points in space. They are created in the same coordinate system as the mesh, which allows their use in processing of the mesh point. Joint rotation information is mostly unused, although the rotation of the 5 spinal joints about the y-axis is used to adjust the calculated rotations of the spinal bones. This will be discussed further in the following section.

5.4.4.2 Creation of Bones

Bones were calculated with positions and rotations which are obtained from the positions of the joints which they connect. Each bone is a connector between

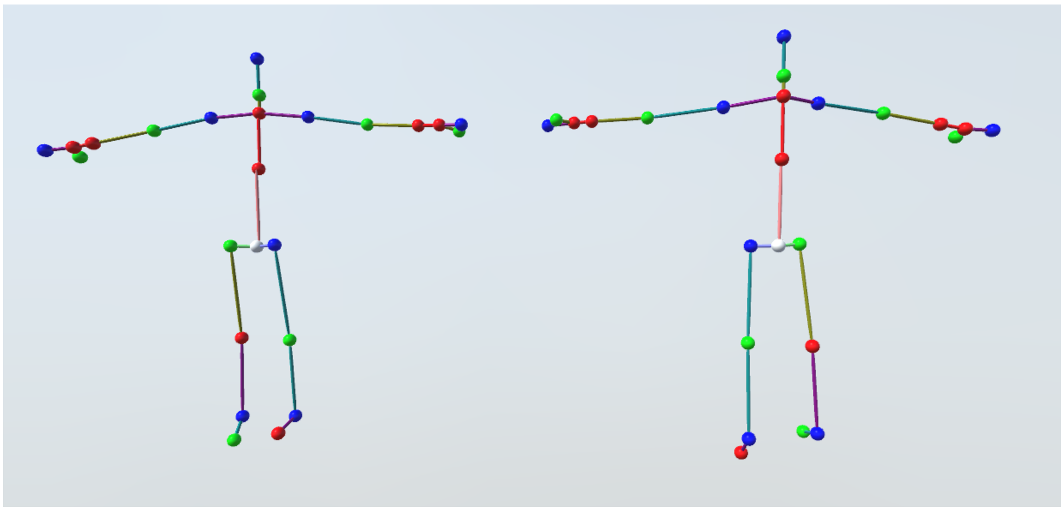


Figure 5.9: Visualisation of Skeleton Data.

two joints. The position of the bone is calculated as the mean of the two joints that it connects, and the rotation is calculated as the bearing from one joint to the other. During development, it was found that while most joint rotations between animation frames could be calculated fairly consistently, the rotations of the bones which connected the spinal joints were being calculated with arbitrary rotations around the y-axis. As a solution, the y-axis components of the rotations of the spinal joints were used to correct their corresponding bones. This approach was not used for the other bones, as their rotations were being calculated consistently by the system. For display purposes, scales for the bones were also created — so that the visible bone objects connected to both of their relative joints. A representation of the resultant skeleton structure can be seen in Figure 5.9.

5.5 Summary

This chapter details the scanning process, the limitations and requirements which influenced the design of those processes, the nature of the data received from the Kinect second generation sensor, the steps required to prepare that data to be turned into a mesh, and the steps required in the process of creating a mesh.

The capture and creation of the skeleton data was discussed. This data will be required for the processing of the mesh, which will be further discussed in Chapter 6, and the animation of the processed me over a network, which will be further discussed in Chapter 7.

CHAPTER 6

PROCESSING THE BODY SCAN MESH

Once appropriate front and back scans are selected by the user, they can be processed into the body scan that will be able to be sent across the network and animated. As discussed in Section 4.3.2, a method was selected that involved the use of skeletal animation over a network to animate a body scan comprised of cage meshes.

To achieve the form of body scan which we wish to animate with this method, our front and back scans must be partitioned, aligned and recombined, gaps between the mesh surfaces must be filled in and overlapping surfaces, and other graphical anomalies, must be removed. Further adjustments and finishing touches are also necessary to create a relatively realistic animation model.

This chapter discusses the necessary steps in processing the body scan data which we have acquired into the form which will allow us to animate it over a network with our chosen method.

6.1 Partitioning the Front and Back Meshes

The initial step in this process is to separate the mesh of the body into parts which are likely to move independently from one another, to obtain cage meshes (similar to the approach used by Li et al. [29]) which can be attached to bones or joint objects and move with them when we animate the mesh over a network. To do this we first identified smaller, very localized groups of points based on their nearest bones or joints. These points were then grouped into larger groups based on which parts of the body were expected to move independently from one another.

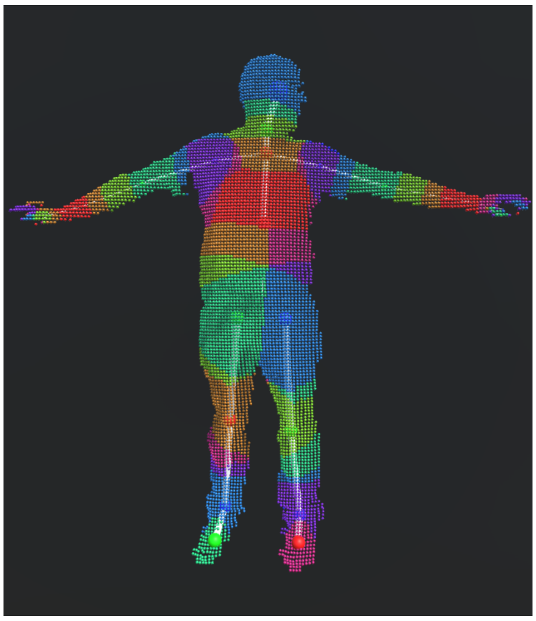


Figure 6.1: Visualisation of vertices with their color matching that of their closest joint or bone.

6.1.1 Classifying Mesh Points by Closest Joint or Bone

For each mesh vertex, the distance to every joint was calculated and the shortest distance was used to find the closest corresponding joint for each vertex. The distances were capable of being scaled based on the relevant joint or bone, but this scaling was unnecessary, as the unscaled groupings were thought to be satisfactory. Figure 6.1 was a display created in the development process to help identify how points should be grouped based on their nearest bones or joints. A point cloud of the mesh vertices was created, and each point was given the color of the nearest joint or bone to the vertex.

6.1.2 Selecting Groups of Joints and Bones

Using this image, groups were chosen based on an impression of which parts of the body were likely to move together, following the motions of a bone or joint. This grouping resulted in the selection of 21 groups. These include three groups along the spine (Head, Upper Torso and Lower Torso) and nine groups each along the left and right sides (Shoulder, Upper Arm, Elbow, Lower Arm, Hand, Upper Leg, Knee, Lower Leg and Foot.) These groups can be seen indicated in Figure 6.2.

6.1.3 Splitting the Mesh Among Groups

Sub-mesh objects were created for each of the chosen groups. Each vertex of the original mesh was copied into a new sub-mesh based on the identified group for

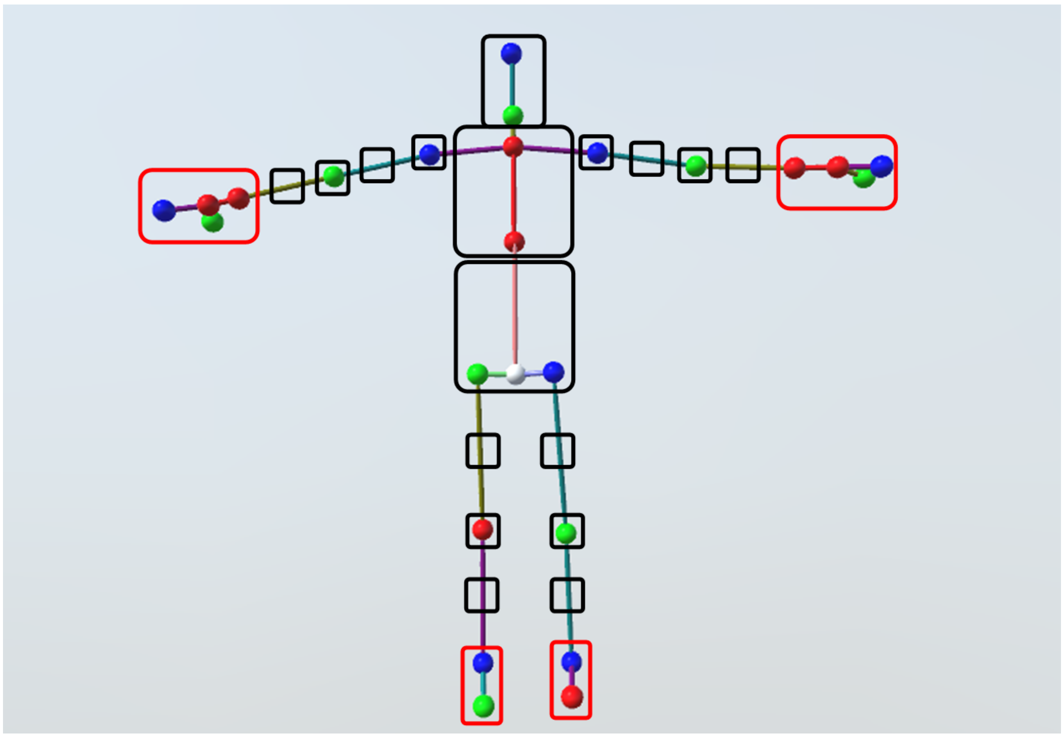


Figure 6.2: Visualisation the skeleton showing which joints and bones correspond to which groups.

that vertex, which was determined by which group the closest joint or bone to that vertex belonged. The corresponding UVs of each vertex were copied to the same sub-mesh.

Triangles for the mesh were sorted into sub-meshes based on which groups the vertices that they connected belonged to. Should all the vertices of a triangle belong to a same group, that triangle would be created in the new sub-mesh, with updated values to refer to the new indices of the vertices of that sub-mesh. Triangles with vertices which belonged to 2 or more groups were not copied into the new sub-meshes, but were stored with references to which sub-meshes, and which vertices within that sub-mesh, that they connected. Edges created by this removal of triangles were also stored for later use, as were references to the vertices which were on these created edges. This triangle and edge information was preserved for the construction of linking meshes, which will be further discussed in Section 6.3.3.

6.2 Creation of a Full Body Scan

Once we have obtained the front and back meshes, and have done some initial preparation, we wish to align and combine these sub-meshes to create a cage mesh model which is capable of being animated.

6.2.1 Previous Data Preparation

Certain minor operations are done to provide or correct data from the Kinect scan. For convenience and compartmentalization of the operations of the system, these steps were done along with earlier processes, as discussed in Chapter 5. They are, however, relevant to the processing of the scan. As such, this section briefly discusses these operations.

6.2.1.1 Edge Detection

Vertices on the edges of the original scan are detected by determining which vertices are not entirely surrounded by non-empty vertices. This is done in the penultimate step of the erosion and dilation method which is used to identify outlying vertices in Section 5.3.1.

6.2.1.2 Skeleton Reversal

The Kinect sensor is unable to detect whether a human body is facing towards or away from the sensor. As such, the skeleton information must be reversed when the mesh and matching skeleton data are selected as the back mesh. This selection should be done during the process described in Section 5.1.2.

The left and right joints and bones of the skeleton are all switched to correctly reflect which side they represent, which allows for simpler matching to their corresponding joints and bones from the front skeleton. The joints representing the ends of the feet are reflected about the joints which represent the ankles, as they are consistently erroneously estimated facing towards the sensor — when in reality they would always be facing away for a back scan. Hand joints are left as they are, as in the necessary distance and chosen pose for scanning, they deliver unreliable results. The joint representing the head also has an offset. However, because this offset is variable, the difference is corrected in the alignment process described in the following section.

6.2.2 Alignment

Once matching group sub-meshes from the front and back scans were created, we could begin the process of joining the front and back sub-meshes into single cage meshes which, together, would make up the full body scan.

Unity's hierarchical structure was used to store these groups within container game objects which are attached to the position and rotation of the joint or bone object that would dictate their movement — this allowed for easier geometric manipulation of the sub-meshes for the alignment process. These containers are used to align the back sub-meshes to the positions and rotations of the front sub-meshes. Alignment is performed using the positions and rotations of the components of the front skeleton, as this skeleton is a more accurate estimation of the skeleton of the body which was scanned than the corrected skeleton from the back scan. The head joint alignment is corrected by calculating the centroid of the vertices for each sub-mesh in the group (front and back) and aligning their position on the x and y plane. The z-offset of the back scan after this adjustment was found to have a consistent value of approximately 0.15, so the position is additionally adjusted by this value.

6.2.3 Elimination of Overlap

Alignment of the front and back sub-meshes showed significant overlap in vertices along the scanned edges, where the Kinect provides less reliable edge data. Before combining these meshes, points which were determined to be overlapping the opposite mesh were removed. This was done by creating a plane which bisected the proposed combined mesh. The normal of this plane was calculated from the difference vector between the centroid of the front mesh (the average position of all the vertices) and the centroid of the back mesh. Since this value was always found to be close to $(0, 0, 1)$, this normal was used for all bisecting planes, to simplify the calculations. The z-coordinate of this plane was determined as the z-coordinate of the average of the two centroids. This meant vertices from the front mesh segment which crossed this plane onto the side of the back mesh segment (easily determined by a comparison of their z values) would be identified as outliers, as would vertices from the back mesh segment which crossed onto the side of the front mesh segment. These vertices were removed, and triangles which referenced any removed vertices were also removed.

During this process, edge vertices (referred to in Section 6.2.1.1) were updated. Edge vertices that were removed as overlapping points were removed from the list, while newly created edge vertices were identified by finding any vertices which were referenced by any removed triangles — as these points would now be on the edge of the sub-mesh.

6.2.4 Combining Mesh Segments

The sub-meshes were then combined into single cage meshes by a Unity function. This created a new combined mesh in the same location as the two sub-meshes, which were still used for processing. The gap between the front and back sub-meshes required the generation of triangles to link the two sub-meshes. A variation of the marching squares method (discussed in Section 5.4.2.1) was used to generate these triangles. However, as the triangles were generated from two ordered lists of non-empty vertices and the mesh was not uniform, only 3 points were selected at a time, and a triangle was always generated. The first two points would be the points from each side which were the last in the order of each list to have been used to create a triangle, and the third point would be whichever point was closest to one of those points.

6.3 Adjustments for Animation

Once the cage mesh model which is capable of being animated has been created, a few further steps are required to supplement the quality of the scan and to make the body scan appear coherent and whole.

6.3.1 Replacement of Hands and Feet

The data for hands and feet gathered from the scan was of a low quality, due to the distance necessary to achieve a full body scan with a single Kinect sensor. Replacement prefabricated 3D objects were used as replacements for both sides. These were free 3D cad objects from `free3d.com`, a website for the sharing of CAD (Computer assisted drawing) files. [4, 35] In order for the linking meshes to have edges to attach to on these objects, the prefabricated objects used the created edges of the group sub-meshes of the hands and feet groups, which were generated when the original mesh was partitioned. Edge points were created at regular intervals and comparable points on the prefabricated 3D objects, and then the created edges from the hands and feet sub-meshes were moved to the closest of the edge points on the prefabricated objects. This would allow the triangles removed in the partition process to still be used in the creation of the linking mesh.

6.3.2 Creation of Full Body Scan Textures

Unity meshes require a single texture, and the combination of the front and back sub-meshes meant that the new cage meshes contained UVs which referenced points on the textures from both the front and back scans. Combining both images into a single texture resulted in poor texture wrapping, as the interpolated values between front and back meshes showed texture data which was being rendered out of place. To achieve a better texture wrapping, textures were split into 5 groups: Head, Torso, Legs, Left Arm and Right Arm. This was done by calculating the bounds from the UV data for the vertices at the upper and lower boundaries of certain regions (Arms, Legs, Torso). These boundaries were then overlapped slightly to provide textures which would not leave the meshes with missing information. The texture segments from the front and back scans were then joined, with the texture from the back mesh texture segments being vertically flipped and placed on top of the front mesh texture segments. Values were then extrapolated to fill in the surrounding empty texture space, to avoid missing texture data being displayed on the rendered mesh. The final textures are displayed in Figure 6.3. The UVs in the cage meshes were also updated to refer to the correct corresponding points on the texture segment of whichever texture group they belonged to.

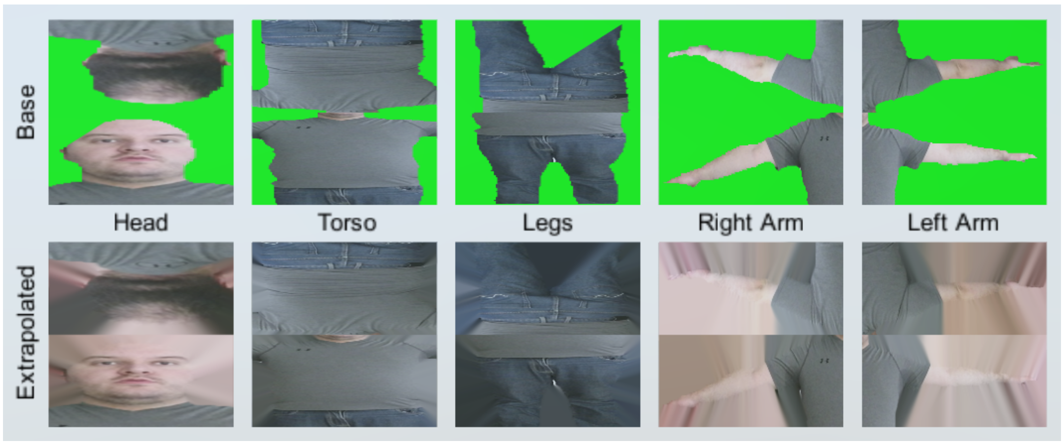


Figure 6.3: Segmented joint textures before (above) and after (below) color extrapolation.

6.3.3 Creation of Linking Mesh

The removed triangles and created edges from Section 6.1.3 were used to generate linking meshes. Five linking meshes, one for each texture group, were created. The positions of the created edge vertices were used as the vertices of this linking mesh and the triangles were created from those which were removed when the mesh was partitioned. The UVs were created from the original mesh UV data, as linking meshes had to refer to the texture segment for their specific group.

The positions of the vertices on the created edges were also used to create empty Unity game objects at the positions of those vertices which lay along the edges of the scan created by the partitioning of the meshes into sub-meshes. These objects were then placed into the same container objects in which the cage meshes had been placed. The purpose of these objects was to provide a reference object which moved when skeletal animation moved the container carrying a cage mesh to a new position and rotation. The new global positions of these objects could then be used to update the position of the vertices of the linking meshes. In this way the moving combined meshes of the body scan would be linked and appear to be a single mesh.

6.4 Summary

In this chapter we discussed how the front and back meshes were broken into sub-meshes which were tied to the movement of certain joints or bones, and how they were combined with each other to make a body scan composed of cage meshes. We also discussed the use of prefabricated objects to replace the hands and feet of the body scan, to supplement the scans overall quality, the processing of the mesh textures, and the linking mesh which was created to tie the cage meshes of the body scan together.

Chapter 7

Transmission and Display of the Body Scan

This chapter details the process of transmitting and animating the final scan over the network. We discuss the steps in serializing the data so that it is capable of being transmitted in a network stream, the network protocols and processes for transmission which were selected, and the process of displaying and animating the scan with the data sent over the network in a navigable, virtual reality environment.

7.1 Serialization

In order to transmit data over a network, we have the initial requirement that this data must be serialized. It was necessary to create functionality to serialize the body scan data while maintaining the created relationships between the data which would allow for animation. Skeleton data was much simpler to serialize, but certain issues needed to be addressed for both components to be able to be serialized for network transmission.

7.1.1 Serialization in C#

C# allows the serialization of classes, provided that these classes are composed of other serializable forms of data. It is possible to serialize to a binary format, which offers the lowest possible bandwidth, and to serialize to XML, which allows for human readability of the data. As low bandwidth is a requirement of this system, and the available data is not inherently geared towards human readability, binary serialization was used. [44]

7.1.2 Data Types to be Serialized

Certain Unity data types, which are heavily used for mesh objects and manipulation of the position and rotation of game objects, are not serializable. Serializable versions of these objects must be created for serialization to be possible of classes or structs which contain these objects. These objects are:

Vector2 Two floats describing a point in 2D space. (x, y)

Vector3 Three floats describing a point in 3D space. (x, y, z)

Quaternion Four floats describing a rotation in 3D space. (x, y, z, w)

Texture2D An image with a width, height and RGBA values for each pixel.

Serializable structs were made for each of the first 3 objects. The Texture2D was converted to an array of bytes for serialization which contained only RGB values for every 3 consecutive elements, and was transmitted alongside width and height data for later reconstruction. Once the component serializable data types were created, we were able to serialize the body scan and skeleton data for our system.

7.1.3 Body Scan Data Serialization

The serialized body scan data contained information on the position and rotation of the container objects, the mesh data for the cage meshes, the mesh data for the linking meshes, the group textures and information for the creation of "edge hooks" necessary for the updating of the linking meshes. The serialized skeleton class contained the following data:

- Positions of container objects. (Vector3, 21 elements.)
- Rotations of container objects. (Quaternion, 21 elements.)
- Cage mesh vertices. (Vector3, array of 21 arrays of variable size.)
- Cage mesh triangles. (Integer, array of 21 arrays of variable size.)
- Cage mesh UVs. (Vector2, array of 21 arrays of variable size.)
- Group textures. (Byte, array of 5 arrays of variable size.)
- Group texture widths. (Integer, 5 elements.)
- Group texture heights. (Integer, 5 elements.)
- Linking mesh vertices. (Vector3, array of 5 arrays of variable size.)
- Linking mesh triangles. (Integer, array of 5 arrays of variable size.)

- Linking mesh UVs. (Vector2, array of 5 arrays of variable size.)
- Edge hook group indices for linking mesh vertices. (Integer, array of 5 arrays of variable size.)
- Edge hook vertex indices for linking mesh vertices. (Integer, array of 5 arrays of variable size.)

The functionality created for the serialization of this data both converts this data into a format which is capable of being serialized and, when deserializing, reverts it to its original format.

7.1.4 Skeleton Data

The skeleton data used for animation was much simpler, containing:

- Positions of joints. (Vector3 type, length of 25.)
- Rotations of joints. (Quaternion type, length of 25.)
- Positions of bones. (Vector3 type, length of 25.)
- Rotations of bones. (Quaternion type, length of 25.)

The bone data is calculated for a debugging display on the computer on which the scan is recorded, and as such is transmitted over the network so that it need not be calculated on the computer used for viewing the scan. Should further reduction in bandwidth be required, recalculating the bones on the viewing computer would halve the amount of data required for this frame. Additionally, not all joints and bones may be relevant, so a selection of necessary components would provide a greater reduction still.

7.2 Transmission

Once serialized, our data was able to be transmitted over a network. The selection of network protocols was necessary, and certain processes were created in order for a scan to be sent from its source to the environment where it would be viewed, as well the stream of frames of skeleton data which would animate that scan.

7.2.1 Network Protocol

Both Unity and C# offer functionality for networking. Unity provides a high-level networking class designed to facilitate multi-player game environment, [67] while C# offers UDP and TCP classes for transmission of data over IP networks.

[43] The high-level functionality of the Unity networking class was not required, as the network transmission requirements of the system were relatively simple. As a result, development of the networking component for this system used the C# classes.

TCP (Transmission Control Protocol) is a connection based protocol where a server-client connection is established and packets of data can be transferred in both directions. TCP ensures ordered data transfer and retransmission of lost data packets, and provides flow and congestion control for the network. UDP (User Datagram Protocol) is a simpler, connectionless protocol where packets of data are sent across the network without a connection. As such, no measures are taken to ensure the packets are received in the order in which they were sent, and packets may be lost or duplicated during transmission. TCP was selected, as the reliability and ordered transfer of the protocol were regarded as necessary for the real-time transfer of the skeleton data for the purposes of animation. While UDP, as a more light-weight protocol, would achieve faster transfers, it is likely the development of an additional protocol to deliver an animation from UDP transferred data would be unlikely to be as efficient as the use of TCP.

7.2.2 Transmission Methods

A process was created for the transfer of the constructed body scan and the continuous transfer of the skeleton data used for animating that body scan. Two computers are connected to a network — the scanning computer, to which the Xbox Kinect sensor is connected, and the viewing computer, to which the Oculus Rift virtual reality head mounted display is connected. The viewing computer listens over the network for an incoming TCP connection on a specified socket. The system then requires the transmission of the body scan, and once this is successfully completed the skeleton data used for animation is continuously sent across the network.

7.2.2.1 Transmission of Body Scan

Once a user has selected the meshes from which the body scan is created, and triggered the processing of the meshes into a body scan, as discussed in Chapter 6, the scan is processed, and the scanning computer attempts to create a connection with the viewing computer. Should the connection be established, the scanning computer transmits the body scan to the viewing computer across the TCP connection. Once the scan has been fully received, the viewing computer will create a body scan from the deserialized body scan data.

7.2.2.2 Transmission of Skeleton Data

Once the viewing computer has received and created the scan, it sends an acknowledgement to the scanning computer and closes the connection once this acknowledgement has been sent. The viewing computer then begins to listen on a separate socket for another incoming TCP connection from the scanning computer.

After receiving the acknowledgement, the scanning computer attempts to create a connection with the viewing computer for the transmission of skeleton data. Once this connection is established, the scanning computer will send a frame of skeleton data every time a new frame of skeleton data is received from the Kinect sensor. Timestamps are included with the skeleton data, but in the current system they are not used, as the ordered data transfer of TCP delivers the frames in the correct order, and the time of transfer is sufficiently small compared to the frame rate of the Kinect sensor that the animation does not lag by varying times between frames. The Kinect sensor produces 30 frames per second, so provided that all Kinect frames are valid, the system will transmit a frame of skeleton data for every frame received from the Kinect.

7.3 Display and Navigation

Once our data has been transmitted across the network, we are required to create a representation of that body scan on the viewing computer and animate that scan with the continuous stream of skeleton data. That animation is then rendered in a navigable virtual environment viewed through a virtual reality head mounted display.

7.3.1 Creation of the Body Scan Display

The processes scan, which is sent over the network, is deserialized in the viewing application, providing the data mentioned in Section 7.1.3. The viewing application then creates containers with the original position and rotation of the joints and bones which control their movement. Using Unity's hierarchy, the cage mesh objects are created as children objects of these containers — which will result in them undergoing the same transformation as those applied to the container objects. These are created from the vertices, triangles and UVs from the deserialized scan data, as well as referencing the group textures.

Link meshes are then created from the link mesh vertices, triangles, UVs, and group textures. Non-rendered, empty game-objects are also created from each vertex in these linking meshes. The deserialized data provides indices linking each of these objects to a cage mesh — these objects are put into the same container as the cage mesh which they reference, and align with all of the vertices on the edge of that cage. The purpose of these objects is to act as "hooks" — moving within the container when its position and rotation is updated, maintaining alignment to the edge of the cage mesh, and providing the new positions for the linking mesh vertices, which shall now be discussed in Section 7.3.2.

7.3.2 Animation

An update of the body scan is triggered by each new frame of skeleton data received over the network. As this data is being continually streamed, it results in an ongoing series of updates which creates an animation.

A frame of skeleton data contains the rotation and position of all the joints and bones from each frame scanned by the Kinect sensor. For the relevant joint and bone data — those which control the movement of the cage meshes — the new joint and bone positions and rotations are applied to the container objects which hold the cage meshes. This translates and rotates the children objects of the container — the cage meshes and the hook objects. For each vertex in each linking mesh, a new position is found from the new position of the relevant hook object. The linking mesh vertices are then updated to their new positions, which deforms the linking meshes into new forms which link all of the cage meshes in their new positions. In this way, the various cage meshes which make up the body are moved, and the linking meshes are updated to maintain the appearance of the body scan as a complete and whole mesh. The quick succession of these updates then effectively creates an animation.

7.3.3 Virtual Environment

A virtual room was created for the final display of the body scan, using various furniture obtained from a free asset pack on the Unity Asset Store. [37] This was done to ease user transition into a virtual environment and establish a staging area where users could evaluate the body scan in a virtual reality context.

7.3.4 Virtual Reality Display

The Oculus Rift virtual reality head mounted display was used, enabling the body scan to be displayed in a Virtual Reality environment. As stated in Section 4.2.2, version 0.6 of the Oculus Rift SDK was used, as later versions did not support the use of integrated graphics, such as was present on the demonstration system. The system nevertheless performed as expected and the display did not lag or incorrectly render any objects in the virtual environment.

7.3.5 User Navigation

Given the difficulty of using a keyboard and mouse while using a virtual reality headset, a navigational system was created for this virtual environment using a joystick controller, where users could control their motion in two dimensions. The interface also allowed for a temporary adjustment of viewing angles, and a reversal of all axis, as a way to extract themselves from viewing errors caused by the software or the headset. Movement of the user's view was restrained to a bounding box which did not reach to the edges of the virtual environment, keeping users in the demonstration space and preventing intersecting views of virtual objects which appeared to be solid.

7.4 Summary

This chapter details the process by which the data for the body scan and the skeleton data are serialized and transmitted across the network. The body scan is transmitted once, while the skeleton is continuously streamed in order to create an animation. This chapter also discusses how the display of the body scan is created from the data sent across the network, how the stream of skeleton data is used to create a network, and how this data is presented in a navigable virtual environment.

Chapter 8

Experimental Design

The system has quantifiable aspects which can be directly measured to ascertain whether they meet the requirements of the stated objectives — such as the bandwidth of the transmission, the time required to process the mesh data into a model of the body and the time required to transfer each frame of skeleton data.

The resultant animation sent across the network requires a more subjective evaluation. Participants not involved in the study can be recruited to provide subjective feedback on the quality of the scan and the animation.

8.1 Quantitative Analysis

The bandwidth used by the system for the transmission of the scan can be compared to the bandwidth that would be required for constant transmission of all the data necessary which is necessary to display a scanned body. The processing times of the set-up can be measured for a number of body models, built from different scan frames, and averaged to see whether the average time and variance fall within acceptable limits. Transmission times can similarly be averaged to determine whether they are as fast or faster than is necessary for the intended purposes of the system.

8.1.1 Bandwidth

The continuous bandwidth required for the system can be determined by measuring the amount of data that is transmitted as the skeleton information used for the animation.

Mesh compression is not used in our system, so we may wish to compare the system to another system, which instead uses a method of directly streaming meshes using mesh compression.

8.1.1.1 Comparison to Direct Streaming of Kinect Data

The amount of data used to transmit the skeleton data can then be compared with the amount of data which would have to be transmitted for all necessary frames of the Kinect data which would be required to construct an equivalent body scan on the receiving device. This comparison would allow us to establish the degree of bandwidth reduction that is achieved. This data would include the BodyIndex, Color and Depth sources from the Kinect sensor, for both the front and back of the body. A comparison can then be made to the data sent across the network for each frame of the animation, to determine to how much less bandwidth the system uses per frame. This can be described by the formula:

$$F = S/K \tag{8.1}$$

Where S is the size of the skeleton data which is sent across the network, K is the size of the Kinect data (for both sides of the body) which would need to be sent across the network for direct streaming of the animation, and F is the ratio of bandwidth reduction for the frame.

The system also has a once off transmission of data for the scan. The bandwidth reduction ratio can be calculated for an entire animation by comparing the sum of the size of the once off scan data and the total size of all skeleton data to the total size of all the Kinect data that would have been transmitted to directly stream the data. The formula for this would be:

$$A = (B + n * S)/n * \Sigma K_n \tag{8.2}$$

Where B is the size (in bytes) of the body scan, n is the number of frames, ΣK_n is the sum of the amount of all Kinect data which would have been transmitted in direct streaming, and A is the ratio of bandwidth reduction for the entire animation.

8.1.1.2 Comparison to Transfer of Time Varying Meshes With Compression

While previous calculations provide us with an impression of the total reduction of data used in the system from the point of scanning to the point of transmission, one might wish to also compare the data reduction of the system to a similar system, such as the system created by Mekuria et al. [40, 41].

Mekuria et al. report a total mesh size per frame of mesh data, which seems to be a good metric for comparison with our system. Given the lower graphical quality achieved by our system compared to that which was achieved in that system, our system should have a per frame transmission size which is suitably smaller in order to make the trade-off worthwhile. [40] Additionally, our system does not reflect changes to the body being scanned in real time in the way that the system created by Mekuria et al. would. Theoretically, our system could periodically update the mesh at fixed intervals to correctly reflect a changing body scan. This would then involve the transmission of a full set of body data at these intervals, and we could then compare the data required to send all frames of mesh data in the system created by Mekuria et al. for that interval against the body scan data sent for the update as well as all the skeleton data sent for that same interval.

8.1.2 Processing Time

Evaluation of the processing time for the system, on the computers used for the study, requires measuring the time taken to compute the mesh, the time taken to send the processed body scan to the receiving system and the time required to send each frame of animation over the network.

The first and second values are subjective, as it is probably a user dependent on what set-up time is unacceptable for the system. Ideally, however, the two processes should not take more than a few minutes each on computers such as those used in the study.

8.1.3 Transmission Time

As the suggested applications for this system in Chapter 1 are, in their simplest forms, intended for the purposes of communication, the requirements for the speed of transfer are less subjective, as transfer times can affect immersion. The International Telecommunication Union recommends Voice Over IP systems should have a latency below 150 ms so as to not negatively impact interaction over the system, and the latency ideally would be below 100ms so that it would be undetectable by users. [22] Should this system become interactive, a similar requirement could be assumed, particularly if an audio component is included. Therefore, the network transmission of the meshes would ideally take place below these values, particularly below 100 ms as other components would possibly contribute their own delays.

8.2 Qualitative Analysis

To evaluate the quality of the system, an approach such as those used by Witmer et al. [73] and Usoh et al. [69] will be used. The system will be demonstrated to users, along with baseline demonstrations for comparison, and questionnaires will be used to established user perceptions of those demonstrations, as well as information about the users which may be relevant to how they perceive the demonstrations.

8.2.1 Methodology

As suggested in Section 2.5, the questionnaires of Witmer and Singer [73] will be used. The questions will be modified to remove questions which are not relevant to this study, and adapted where possible. With the criticisms of the Witmer and Singer's presence questionnaire in made by Usoh et al. in mind, additional questions will also be added by the researcher in an attempt to supplement the existing questions in use. Volunteers will be recruited to participate in the study. They will initially sign consent forms, and then complete a version of the ITQ to determine their immersive tendencies. They will then view three demonstrations, which will be discussed in Section 8.2.2, and after each demonstration will complete a version of the presence questionnaire to determine their sense of presence for each demonstration.

8.2.2 Demonstrations

Three demonstrations were selected for this study, with the aim in mind that users were comparing recording and playback solutions for body scans, rather than comparing the VR experience to reality, as this was not to be an evaluation of the level of VR hardware used for the system, only of the comparative quality of the body scans in the context of the environments in which they were presented.

The evaluations consist of a 2D demonstration, a 3D demonstration and a VR demonstration. All demonstrations will be created from the same Kinect recording played back from the Kinect Studio utility, and all participants will view the same recording. The recording will be of the researcher standing in place and doing a simple set of motions. The 3D and VR demonstrations will be interactive, while the 2D demonstration will not be.

The order will not be randomized, as the questionnaire answers are meant to be a comparison of the users experience of the demonstration. Additionally, the 2D demonstration's comparison to the other demonstrations is not as important as their comparisons to each other, allowing the first questionnaire to serve as a method for the participants to become acclimatized to the process.

8.2.2.1 2D Demonstration

The 2D demonstration displays only the RGB color video feed from the Kinect sensor, using the available example program from the Microsoft Kinect SDK. The background of the image will be masked and left blank. This can be seen in Figure 8.1. This demonstration will be displayed on a laptop screen, and there will be no navigation for this system.

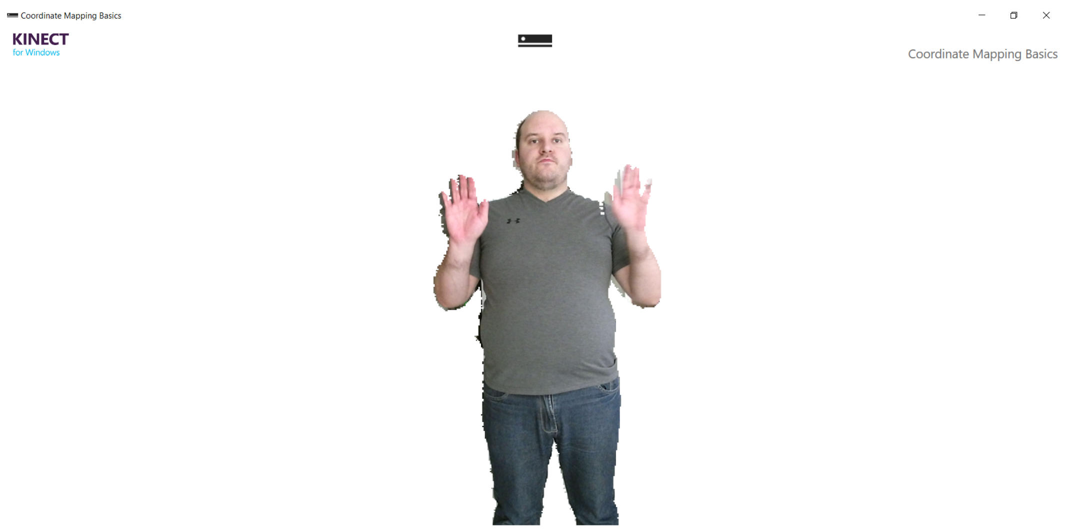


Figure 8.1: Screenshot of the 2D Demonstration.

8.2.2.2 3D Demonstration

The 3D demonstration displays an updating mesh constructed directly from the Kinect data streams as seen in Figure 5.8, with the representation of the skeleton as seen in Figure 5.9 created underneath, and will be viewed in a white 3D box in the Unity game environment. This demonstration will be displayed on a laptop screen, and users can use the arrow keys on the keyboard to move backwards, forwards, left and right. The viewing angle of this demonstration will not be adjustable, as only the front half of the mesh is directly available from the Kinect sensor. This demonstration is shown in Figure 8.2.

8.2.3 VR Demonstration

The VR demonstration is a virtual environment in which the transmitted body scan of the system built for this study is shown. It displays the processed body scan, which will be processed on the laptop and sent across the network to the demonstration computer, where a body scan will be created in the virtual room in the Unity game environment and animated by the streaming skeleton data from the laptop.

This demonstration is displayed on the Oculus VR headset, and a duplicate non-VR view will be displayed on a nearby screen, enabling the researcher to follow what the users are seeing. This demonstration can be seen in Figure 8.3.

The system will update the view based on the position and orientation of the user's head, and users will be given a game controller to be able to additionally adjust their viewing angle and their location, to accommodate for limitations in motion created by the Oculus Rift cables and the demonstration space.

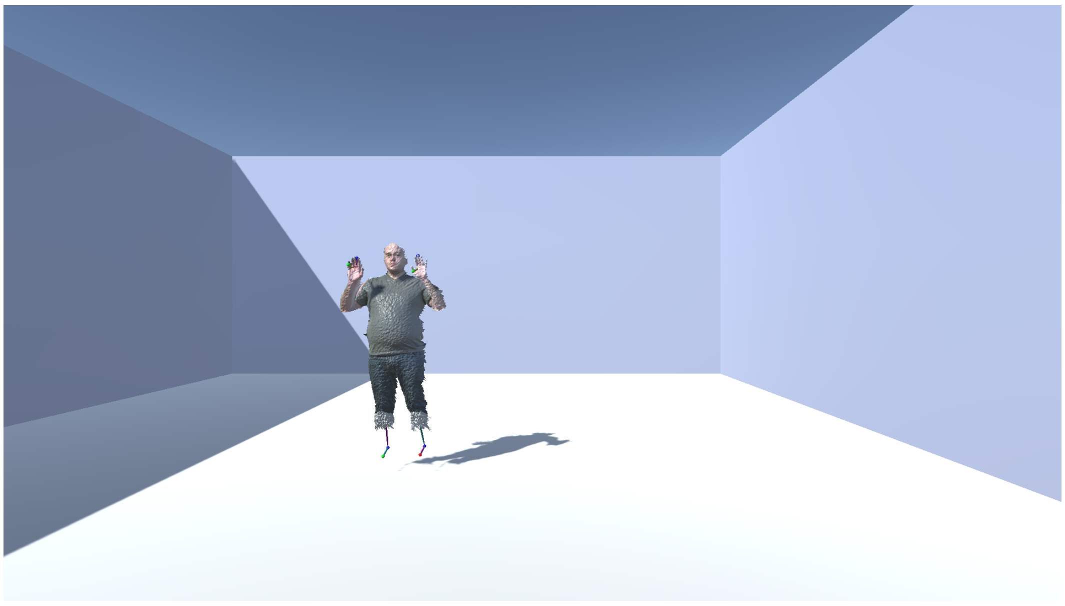


Figure 8.2: Screenshot of the 3D Demonstration.

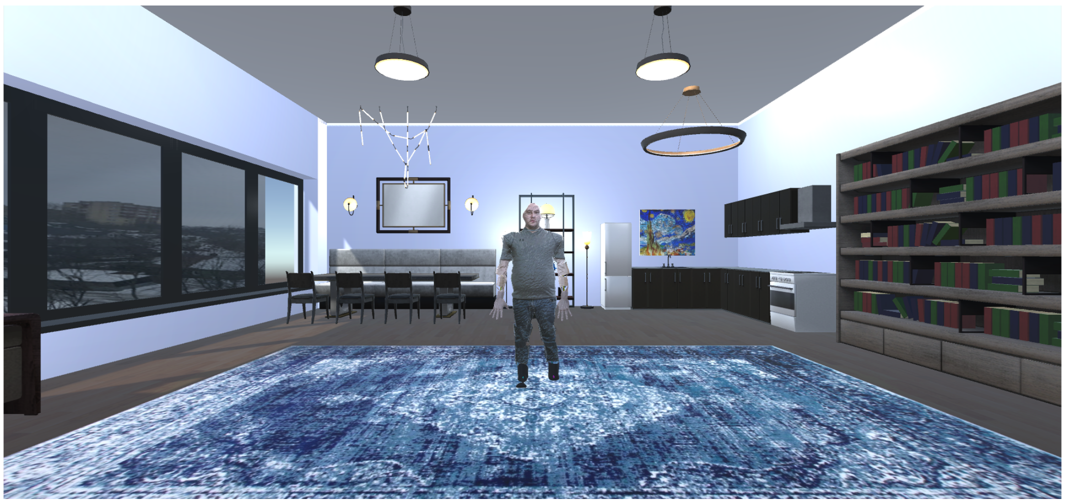


Figure 8.3: Screenshot of the VR Demonstration, as seen on monitor view.

8.2.4 Questionnaires

As stated in Section 8.2.1, questionnaires similar to those used by Witmer and Singer [73] were used. As no answer scales for the questions were provided, answers were created, and questions were adapted so that these answers were uniform where possible, allowing for faster answering from users and minimal explanation from the researcher.

8.2.4.1 Immersive Tendency Questionnaire

The ITQ (Immersive Tendencies Questionnaire) establishes the propensity of an individual to become involved in activities such as work tasks and entertainment. It also attempts to establish a rough impression of the individuals current mental state, and obtain an indication of their degree of their preferences of various types of entertainment — video games, sports, reading and television. Some questions can be grouped into categories, which are represented by letters in the group column. These categories are:

Focus (F) Tendency to maintain focus on current activities.

Involvement (I) Tendency to become involved in activities.

Games (G) Tendency to play video games.

Miscellaneous (M) Questions which were not grouped, and do not necessarily indicate the same factors.

The first three categories can be grouped, and used as values for comparison to the presence questionnaire data. The ITQ questions were largely kept, albeit in their adapted forms, although it should be noted that certain questions will most likely not be used, due their very low correlation with their relevant groups. (Questions 4, 11, 12, 19, 24 and 27.) While these questions are asked in the hope that they will be able to provide extra insight into answers given in the presence questionnaire, they do not evaluate the experiences of the demonstrations in any way.

#	Question	Low Score	High Score	Group
1	Do you ever get extremely involved in projects that are assigned to you by your boss or your instructor, to the exclusion of other tasks?	Never	Very Often	M
2	How easy do you find it to switch your attention from the task in which you are currently involved to a new task?	Very Difficult	Very Easy	M
3	How frequently do you get emotionally involved (angry, sad, or happy) in the news stories that you read or hear?	Never	Very Often	M
4	How well do you feel today?	Very Bad	Very Good	M
5	How easy is it for you to become deeply involved in movies or TV dramas?	Very Difficult	Very Easy	F
6	Do you ever become so involved in a television program or book that people have problems getting your attention?	Never	Very Often	I
7	How mentally alert do you feel at the present time?	Not At All	Very Alert	F
8	Do you ever become so involved in a movie that you are not aware of things happening around you?	Never	Very Often	I
9	How frequently do you find yourself closely identifying with the characters in a story line?	Never	Very Often	I
10	Do you ever become so involved in a video game that it is as if you are inside the game rather than moving a joystick and watching the screen?	Never	Very Often	G
11	On average, how many books do you read for enjoyment in a month?			M
12	What kind of books do you read most frequently?			M
13	How physically fit do you feel today?	Very Unfit	Very Fit	F
14	How good are you at blocking out external distractions when you are involved in something?	Very Bad	Very Good	F
15	When watching sports, do you ever become so involved in the game that you react as if you were one of the players?	Never	Very Often	M

16	Do you ever become so involved in a daydream that you are not aware of things happening around you?	Never	Very Often	I
17	Do you ever have dreams that are so real that you feel disoriented when you awake?	Never	Very Often	I
18	When playing sports, do you become so involved in the game that you lose track of time?	Never	Very Often	F
19	How easy is it to disturb you when you are working on a task?	Very Difficult	Very Easy	M
20	How well do you concentrate on enjoyable activities?	Very Poorly	Very Well	M
21	How often do you play arcade or video games? (VERY OFTEN should be taken to mean every day or every two days, on average.)	Never	Very Often	G
22	How well do you concentrate on disagreeable tasks?	Very Poorly	Very Well	M
23	Have you ever gotten excited during a chase or fight scene on TV or in the movies?	Never	Very Often	F
24	To what extent have you dwelled on personal problems in the last 48 hours?	Not at All	Very Extensively	M
25	Have you ever gotten scared by something happening on a TV show or in a movie?	Never	Very Often	I
26	Have you ever remained apprehensive or fearful long after watching a scary movie?	Never	Very Often	I
27	Do you ever avoid carnival or fairground rides because they are too scary?	Never	Very Often	M
28	How frequently do you watch TV soap operas or docu-dramas?	Never	Very Often	M
29	Do you ever become so involved in doing something that you lose all track of time?	Never	Very Often	F

Table 8.1: Immersive Tendencies Questionnaire.

8.2.4.2 Presence Questionnaire

The PQ (Presence Questionnaire) was more heavily adapted, and questions involving audio and haptic sensory feedback to users were not included, as the system lacks the relevant components. Additionally, questions based on tasks and task performance were removed, as users were not required to complete tasks in the demonstrations, as the demonstrations were only meant to be viewed and, where applicable, navigated. Questions referring to the environment or objects were, in both cases, altered to refer to the body scan. This resulted in questions which were essentially duplicates, and these duplicates were removed. Lastly, four additional questions were added to the questionnaire by the researcher, to directly ask participants about their general impression of the scan, as well as the accuracy of shapes, detail and depth presented in each demonstration.

The data is grouped in two ways, based on the factors and subscales presented by Witmer and Singer. [73]. The first group, based on the factors, are in the G1 column of Table 8.2, and are described as follows:

Control (C) The degree of control over the environment experienced by the user.

Sensory (S) The degree of consistency and accuracy of sensory input experienced by the user.

Distraction (D) The degree to which distracting factors were experienced by the user.

Realism (R) The degree of realism of the virtual environment experienced by the user.

The second group, based on the sub-scales, are in the G2 column of Table 8.2, and are described as follows:

Control and Involvement (C) The extent to which the user felt they were involved in the demonstration, and had control.

Natural (N) The extent to which the user felt the environment in the demonstration was natural.

Resolution (R) The extent to which the users could discern detail in the demonstration.

Interface Quality (I) The extent to which the interface effectively displayed and allowed navigation to users in the demonstration.

In both columns, the letter *M (Miscellaneous)*, indicates that the question was not given a factor or subscale assignment in Witmer and Singers questionnaires, and do not indicate comparable factors in presence.[73] The last four questions, added by the researcher, are not assigned categories either, and the G1 and G2 columns are left blank in the table for these questions.

#	Question	Low Score	High Score	G1	G2
1	How natural did the recording seem (to you) to watch?	Not at All	Completely	C	N
2	How completely were all of your senses engaged?	Not at All	Completely	S	M
3	How engaging was the visual aspect of the recording?	Not Engaging	Very Engaging	S	C
4	How aware were you of events occurring in the real world around you?	Very Aware	Completely Unaware	D	M
5	How aware were you of the device used to display the recording?	Very Aware	Completely Unaware	D	M
6	How compelling was the sense of the recording being a person in the room with you?	Person was On A Screen	Person was In The Room	S	C
7	How inconsistent or disconnected was the information coming from all your senses?	Very Disconnected	Very Connected (No discrepancies)	R	M
8	How completely were you able to visually study the person in the recording? (From all angles/ vantages)	Not At All	Very Completely	S	C
9	How closely were you able to visually study the person in the recording?	Not At All	Very Closely	S	R
10	To what extent did you feel confused or disoriented after viewing the recording?	Not At All	Very Disoriented/Confused	R	M
11	How involved were you in the experience of viewing the recording?	Not At All	Very Involved	M	C
12	How quickly did you adjust to the experience of viewing the recording?	Adjustment Was Very Slow	Adjustment Was Instant	C	C
13	How much did the display distract you from viewing the recording?	Display Was Very Obvious	Didn't Notice Display At All	D	I

14	How well could you concentrate on viewing the recording rather than navigating through the recording for different views?	Focused Completely On Navigation	Focused Completely On Viewing	D	I
15	Were you involved in viewing the recording to the extent that you lost track of time?	Completely Lost Track Of Time	Fully Aware Of Time Passing	M	C
16	To what extent did the body movements of the person appear to be visible as if the person was right in front of you?	Body Movements Not Clear	Body Movements Very Clear		
17	How accurate would you say the presentation of shapes and forms in this recording is? (Does anything look misshapen)	Very Inaccurate	Very Accurate		
18	How accurate would you say the presentation of detail in this recording is? (Does anything look blurred or obscured)	Very Inaccurate	Very Accurate		
19	How accurate would you say the presentation of depths in this recording is? (How easy is it to determine the relative distances of objects)	Very Inaccurate	Very Accurate		

Table 8.2: Presence Questionnaire.

8.2.5 Analysis and Display of Questionnaire Answers

Tables showing the results of the ITQ will be shown in Appendix B and tables showing the results of the PQ will be shown in Appendix C.

Specific groups of ITQ data will be plotted against specific groups of PQ data in an attempt to see whether a users presence results were influenced by their immersive tendencies. Any relevant correlation found will be shown. To allow for a quick visual comparison between the answers received for different demonstrations, violin plots will be created for each question. To approximately ascertain whether users were able to differentiate well between the 2D, 3D and VR demonstrations, we require a visual representation that allows us to compare the true labels of the survey responses with labels that have been generated by an unsupervised clustering algorithm to establish how well the survey responses for each demonstration cluster together.

8.2.5.1 Violin Plots

Violin plots are a combination of box plots and density traces (or smoothed histograms), which show the median value, the interquartile range, adjacent values

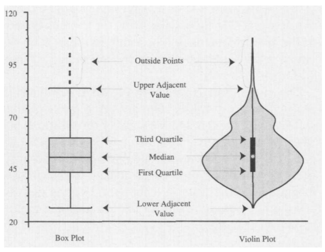


Figure 8.4: Example of a Violin Plot Extracted from [16].

and outlying values. [16] Violin plots will allow the reader to make a quick visual comparison between answers to the same question for different demonstrations. An example of the violin plot can be seen in Figure 8.4.

8.2.6 K-means Clustering

We will reduce the dimensionality of the data to two dimensions using principal component analysis. We can then plot a point for each survey response using these two dimensions. Thereafter, a k-means clustering algorithm will be applied to the reduced data to group the survey responses into 3 clusters — where each cluster corresponds to one of the three demonstrations. The accuracy of these predicted labels will give us insight into how clearly participants were able to differentiate between the 3 demonstrations. [36]

8.3 Summary

Quantitative analysis of the system will involve a comparison of the bandwidths required to send the processed and unprocessed scans across the network, a measurement of the average processing time of a set-up of the body model construction for the system, and a measurement of the average transmission time for each frame of skeletal data used for the animation.

Qualitative analysis will question a number of participants on the comparative quality of the VR experience to a 3D and a 2D experience of the same scan sequence. Should the system meet the goals of the quantitative analysis, and the qualitative analysis shows that the VR demonstration compares favourably to the other demonstrations, the system will be considered a successful proof of concept for this approach to low bandwidth transmission of a body scan animation. These results will be presented in Chapter 9.

CHAPTER 9

RESULTS AND ANALYSIS

Demonstrations were conducted with voluntary participants from students and staff of the University of Stellenbosch. During the demonstrations, values for the bandwidth and processing times for both the set-up and the real-time animation data were measured and then averaged. These were then compared to the calculated theoretical values for the bandwidth required to transmit the full set of scan data across a network, and the ideal times specified in Chapter 8.

9.1 Results of Quantitative Analysis

Using measurements from the system during demonstrations, or under similar conditions, we are able to measure factors for our qualitative analysis.

9.1.1 Bandwidth

We calculate the bandwidth reduction as discussed in Section 8.1.1.1. Where necessary, measurement values were averages calculated from the debugging output of 10 body scans created on the second day of demonstrations.

9.1.1.1 Comparison to Direct Streaming of Kinect Data

We are able to calculate the bandwidth of transmission for the original scan data by calculating the total size of all the frames necessary for the construction of the body scan: The RGB frame of colour values (of 4 bytes each) with a resolution of 1920 by 1080, and two frames of resolution 512 by 424: The depth frame (where depth is a 2 byte value) and the body index frame (a 1 byte value representing the presence of a body pixel.)

An RGB frame:

$$1920 \times 1080 \times 4bytes = 8249400bytes \approx 8,25\text{MB} \tag{9.1}$$

A Depth frame:

$$512 \times 424 \times 2bytes = 434176bytes \approx 0,22\text{MB} \tag{9.2}$$

A Body Index frame:

$$512 \times 424 \times 1byte = 217088bytes \approx 0,22\text{MB} \tag{9.3}$$

Collectively, the total size of the unprocessed Kinect data (per frame) is:

$$8249400 + 434176 + 217088 = 8900664bytes \approx 8,9\text{MB} \tag{9.4}$$

Direct transmission of this data would require a bandwidth sufficient to transmit 60 such frames in one second: 30 frames per second (the frame rate of the Kinect sensor) for both the front and back sets of scan data.

$$8\frac{bits}{byte} \times 8900664\frac{bytes}{frame} \times 60\frac{frames}{second} = 4272318720bps \approx 4,27\text{Gbps} \tag{9.5}$$

Which is an exceedingly high bandwidth requirement and completely impractical for most applications.

Table 9.1 contains data the average data size of the serialized scan and skeleton objects recorded from the system during the course of the demonstrations. The constructed body scan in this system is somewhat smaller than the original Kinect data, at a value of 10,96 MB. The unprocessed scan data for this would be approximately 17,8 MB.

Data	Average Size
Serialized Body Scan	10 960 422 bytes
Serialized Skeleton	3 558 bytes

Table 9.1: Mean Sizes for Serialized Body Scan and Skeleton Objects.

However, this scan is only transmitted once in the constructed system. The ratio of reduction in data achieved per frame (discounting the initial body scan transmitted) can be calculated by:

$$F = S/M = 3558/(8900664 \times 2) \approx 0.00020 \approx 0.02\% \tag{9.6}$$

This amounts to a reduction of approximately 99.98% in the data being sent across the network, by relying on skeletal animation instead of direct mesh transmission.

For the reduction in data for an animation as a whole, we use the animation ratio formula from Section 8.1.1:

$$A = (B + n \times S)/(n \times M) = (10960422 + 3558 \times n)/(8900664 \times 2 \times n) \tag{9.7}$$

Table 9.2 shows that as the number of frames in the animation increases, the effective reduction of data sent across the network increases. At 30 frames — which is one second of animation — the bandwidth reduction is approximately 98%.

Frames in Animation	1	2	5	10	100	1000
Data Reduction Ratio	0.616:1	0.308:1	0.123:1	0.062:1	0.006:1	0.001:1
Reduction Percentage	38,4%	69,1%	87,7%	93,8%	99,4%	99,9%

Table 9.2: Data Reduction Ratio and Reduction Percentage for number of frames in animation.

9.1.1.2 Comparison to Transfer of Time Varying Meshes With Compression

As stated in Chapter 8, a better impression of how well this technique performs can be achieved by a comparison to a system which has similar aims. Mekuria et al. transmit a compressed mesh data with a per frame size of 1446 kilobytes. These meshes are full meshes constructed from multiple Kinect sensors. [40] While the quality of the system is presumably greater than the resultant skeletal animated body scan of the system of this study, the study per frame transmission of 3558 bytes is a magnitude of order smaller than this.

$$3558/(1446 \times 1024) \approx 0.0024 \approx 0.24\% \tag{9.8}$$

This system achieves a per frame size that is 99.76% smaller than that achieved by Mekuria et al., a substantial reduction in the amount of bandwidth required for transmission, which as a trade off for reduced quality could be considered worthwhile.

Similarly, we can also compare how this reduction is offset by the requirement of transmission the original scan data, as we did in Table 9.2. Table 9.3 shows a comparison of the amount of data required to transmit a body scan and n frames of skeleton data, compared to the amount of data required to transmit n frames of mesh data as seen in the system created by Mekuria et al. [40].

Frames in Animation	1	2	5	10	100	1000
Data Reduction Ratio	7.405:1	3.703:1	1.483:1	0.743:1	0.076:1	0.010:1
Reduction Percentage	-640.46%	-270.5%	-48.28%	25.74%	92.36%	99.02%

Table 9.3: Data Reduction Ratio and Reduction Percentage for number of frames in animation when compared to the mesh data size from the system created by Mekuria et al. [40]

For 1, 2 and 5 frames, we see that sending a body scan and skeleton data frames require more data than the equivalent amount of frames from the system created by Mekuria et al. At 10 frames, however, the data sent across our system is less than that sent across in that system. [40].

The practical application of this comparison would be a theoretical version of our system which periodically updates the body scan at fixed intervals. The system does not currently do this, and measures would be necessary to avoid occlusion and obtain appropriate meshes, but our main limitation would be the required times for processing and transmitting the body scan.

From system measurements which will be presented in Section 9.1.2, we determined that combined times required to process and transmit the body scan range from approximately 80 to 115 seconds. The shortest update interval we could then achieve, to the nearest minute, would be 2 minutes. The Kinect sensor produces 30 frames of scan data per second. Presuming that all frames are valid and used in transmission, we have a total of 3600 frames for 2 minutes of transmission.

For our system, we can calculate the total amount of data required to be transmitted by adding the data size of body scan (which is sent once) to the data size of the 3600 frames of skeleton data.

$$10960422 + (3558 \times 3600) = 23769222 \text{bytes} \tag{9.9}$$

We can then calculate the total amount of data required for 2 minutes of animation in the system created by Mekuria et al., which would be 3600 frames of mesh data at 1446 kilobytes each.

$$1446 \times 1024 \times 3600 = 5330534400 \text{bytes} \tag{9.10}$$

We are then able to compare these values to determine if our system uses less data in transmission in this theoretical form than that created by Mekuria et al.

$$23769222/5330534400 \approx 0.0045 \approx 0.45\% \tag{9.11}$$

From this, we can see that a version of our system that periodically updates the mesh every 2 minutes would require per frame transmission sizes that are 99.55% smaller than the system created Mekuria et al. Obviously, Mekuria's system would still be able to update meshes during the course of animation, and as such be more responsive at reflecting changes to the mesh by a factor of 3600.

9.1.2 Processing and Transmission Time

The other factor to be measured by the system was time required for processing and network transmission. These are detailed in Table 9.4.

Action	Min. Time	Max Time	Mean Time
Processing Mesh Data	52.4 s	67.2 s	54.1 s
Transmitting Body Scan Data	27.3 s	47.8 s	42.2 s
Transmitting Skeleton Data	13 ms	89 ms	32.4 ms

Table 9.4: Maximum, Minimum and Mean times for System Processing and Network Transmission.

The combined set-up time of the network has a combined value of 115 seconds, nearly two minutes. This waiting time for the system is within acceptable limits, and can be further reduced with improved processing of the mesh data.

The one-way transmission time of frames for the data is ideal, as this sits safely below the 100 ms limit recommended by the ITU for VoIP applications, and provides enough time for 30 frames of data to be transmitted per second.

9.2 Results of Qualitative Analysis

Participants were recruited at the University of Stellenbosch from the MIH Media Lab in the Engineering Faculty, and the Information Science department in the Arts and Social Sciences Faculty.

38 participants were recruited to the study. All participants were older than 18 years, fluent in English, and signed a consent form to take part in the research. It was explained to the participants that they did not waive any rights by agreeing to participate in the research, and they could stop the process at any time.

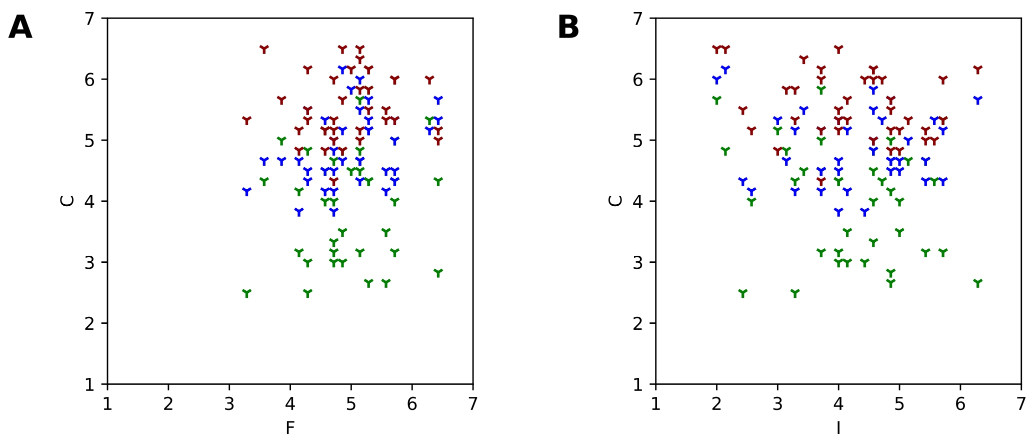


Figure 9.1: Plot of Control/Involvement group mean scores for PQ against Focus mean scores (A) and Involvement mean scores (B) for the ITQ.

Each participant completed an ITQ (Immersive Tendencies Questionnaire), as detailed in Section 8.2.4.1. The researcher then introduced the participants to each demo, allowed them to view and interact with the demo (where applicable) and then the participants completed the PQ (Presence Questionnaire), as detailed in Section 8.2.4.2, for each demo.

The researcher was available to explain the intent of questions to participants, but did not instruct them how to answer, except for the specific case where a question (number 14 in the PQ) was not applicable to the first demo, and participants were told they could provide any answer, as the results of that particular question would not be recorded. No participants asked to stop the demonstrations. The results for the ITQ will be presented in Appendix B and those for the PQ will be presented in Appendix C.

9.2.1 Analysis

For analysis of the data, the ITQ questions and PQ questions were placed into various groups to determine group scores. Data between the ITQ answers and PQ answers, as well as between the different answers from PQs for each demonstration were compared.

9.2.1.1 Comparison of PQ with ITQ

Various ITQ factors were compared against PQ results, but strong correlations were not found. Figure 9.1 shows the plotting of means scores of users answers in the Control/Involvement sub-scale group in the PQ against the mean scores of their Focus sub scale group and Involvement sub scale group answers in the ITQ. The clustering is fairly weak, and no clearly discernible groupings can be seen.

9.2.1.2 Overview of PQ Answers

The answers to the presence questionnaire are presented in Figure 9.2 as a violin plot, showing the medians (white lines), upper quartile and lower quartile values (upper and lower limits of black boxes), minimum and maximum values (edges of the curved shapes) and density of the data (thickness of the curved shapes.)

As can be seen in this plot, the VR received higher rating in all questions except one, question 15, which is inversely indicative of presence. (A score of one indicates the user completely lost track of time during the demonstration, while a score of 7 indicates users were completely aware of time passing.)

These questions are grouped into the factor groups and subscale groups and their means are shown. The means for questions without groups will also be displayed.

9.2.1.3 Factor Groups

Witmer and Singer described certain main factors that each question could be assigned to, which appear to correspond to the factors they propose contribute to a higher degree of presence in a virtual experience. These factors detailed:

- The extent to which the user felt control over the system.
- Their level of distraction during the demonstration.
- The degree of realism they experience.
- To what degree the demonstration engaged their senses.

Table 9.5 shows that all four factors scored more strongly in the 3D demonstration than in the 2D demonstration, and more strongly in the VR demonstration than in the 3D demonstration. Of particular note is the mean scores for the realism factor, which shows a much greater score for the VR demonstration than for the 3D and 2D demonstrations. The sensory factor mean score was also significantly higher than the 3D and 2D demonstrations.

Factors	Control	Distraction	Realism	Sensory
2D Demo	4.6	3.7	2.8	3.4
3D Demo	4.8	4.1	2.9	4.5
VR Demo	5.2	4.7	4.4	6.2

Table 9.5: Mean Scores for the Control, Distraction, Realism and Sensory Factor Group questions.

Of course, such results may be due to the increased immersion associated with use

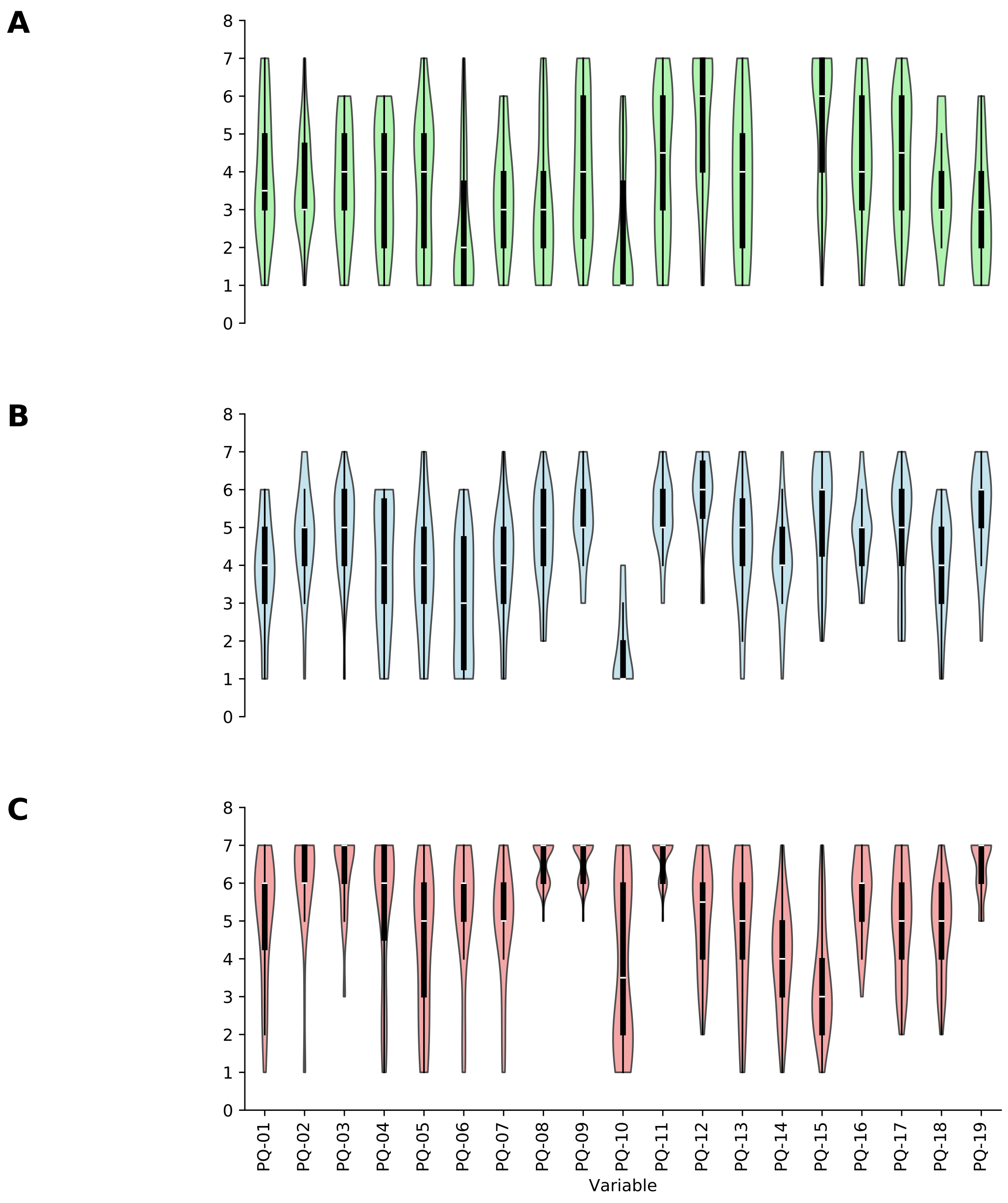


Figure 9.2: Violin Plot showing the distribution and density of PQ answer data for the (A) 2D Demonstration, (B) 3D Demonstration and (C) VR Demonstration.

of a VR device, but it can be taken as a strong indication that the scan transmitted across the network and displayed on the VR headset was not having a negative effect on the sense of immersion felt by users in the virtual environment, and as the sensory group contained many questions about the scan itself, the body scan system appears to have been regarded favourably in the comparison.

Other Questions	PQ-11	PQ-15
2D Demo	4.2	5.4
3D Demo	5.3	5.3
VR Demo	6.7	3.4

Table 9.6: Mean Scores for the questions with no Factor Group.

Table 9.6 deals with the 2 questions that were not placed into factor groups. Question 11 asks users how involved the were in the experience of viewing the recording, while question 15, which as pointed out in Section 9.2.1.2, is an inversely indicative question asking users to what extent they lost track of time.

The answers to 11 seem to show that users felt that they were more involved in each successive demonstration, which is understandable given the increased control and more sensory information available in each experience.

The answers to 15 (where low values indicate losing track of time) suggest that the 2D and 3D demonstrations, which had higher mean scores of approximately 6, did not cause users to lose track of time to a great extent. The VR demonstration, with a mean score of approximately 3, was significantly more likely to cause users to lose their sense of time, suggesting a higher level of involvement.

9.2.1.4 Sub-scale Groups

Witmer and Singer also described a set of sub-scales which questions could be assigned to, and these values appear to be more concerned with a user impressions of a virtual experience, rather than their sense of presence and immersion. These factors detailed the level of Involvement and Control felt by a user in the experience and their impression of the interface with which they accessed the experience. Other sub-scale groups were present but were removed, either because the system had no relevant components for those groups (Auditory and Haptic) or other questions had been removed leaving only one question in that group (Natural and Resolution), which were then moved to the table of single questions.

Sub-scales	Control/Involvement	Interface Quality
2D Demo	4.0	3.8
3D Demo	4.9	4.3
VR Demo	5.9	4.5

Table 9.7: Mean Scores for the Control/Involvement and Interface Quality Sub-scale Group questions.

The results as shown in Table 9.7 indicate that both questions for both factors received higher scores for the 3D demonstration than those of the 2D demonstration, and higher scores for the VR demonstration than those of the 3D demonstration.

The differences, however, for interface quality were low, and it should be noted that during the demonstrations many users commented that the lack of an ability to adjust the viewing angle in the 3D environment hampered their ability to navigate, and that the navigation in the VR environment was somewhat disorientating.

Other Questions	PQ-01	PQ-02	PQ-04	PQ-05	PQ-07	PQ-09	PQ-10
2D Demo	3.9	3.7	3.7	3.6	3.3	4.2	2.3
3D Demo	3.8	4.7	4.0	3.8	4.0	5.3	1.8
VR Demo	5.2	6.0	5.4	4.4	5.0	6.6	3.9

Table 9.8: Mean Scores for the questions with no Sub-scale Group.

In Table 9.8, Question 1 was the only remaining question for the natural factor, and appears to show that the users found the 2D demonstration more natural than the 3D demonstration, but that the VR demonstration was more natural than both.

Question 9 was the only remaining question in the Resolution category, and asked the users how closely they were able to study the person in the recording. The VR demonstration scored comparatively well in this question, but it is unclear whether this is indicative of scan quality on its own, as some user comments suggested that they may have interpreted this question to be about viewing distances that they could achieve in the virtual environment, or about the quality of the head mounted virtual reality display used for the demo.

The other questions are adequately represented in the factor groups above, but question 10 has a lower value for "confusion and disorientation" for the 3D demonstration than the 2D demonstration. Some users commented after the demonstrations that they had answered question 10 inconsistently in the first demo, as they thought the question was asking about their confusion at the processes employed for the demonstration, and not directly to the aspects of the demonstration itself. That the mean score for this question in the third demo was higher despite increased familiarity with the process indicates that users were probably experiencing a degree of actual confusion/disorientation as a result of being in a virtual reality environment.

9.2.1.5 Researcher Questions

Additionally, a set of four alternative questions were created by the researcher, which directly questioned users on their general impression of the body scan, and their impressions of the system's accuracy in representing shape, detail and depth.

Impression	General	Shape Accuracy	Detail Accuracy	Depth Accuracy
2D Demo	4.4	4.4	3.6	2.9
3D Demo	4.9	4.8	4.1	5.4
VR Demo	5.7	5.0	4.9	6.4

Table 9.9: Mean Scores for Researcher System Impressions Questions.

Table 9.9 shows the mean scores of these questions, which also indicate a similar pattern of the 3D demonstration obtaining higher mean scores than the 2D demonstration, and the VR demonstration obtaining higher mean scores than the 3D demonstration.

9.2.1.6 Clustering of Survey Responses

The dimensionality reduction of the data and identification of the dimensionally reduced data points was conducted, as discussed in Section 8.2.6. Figure 9.3 shows the data plotted on a scatter plot coloured to represent their true labels (2D = Green, 3D = Blue, VR = Red) in A, while B shows the plotted points grouped by k-means clustering.

Table 9.10 shows the confusion matrix for the classification for the classification of this data, where rows represent the predicted values and the columns represent the actual values. This matrix shows that the classification tends to show a high degree of accuracy when identifying separate surveys, but has a higher degree of confusion when distinguishing the 2D demonstration survey responses from the 3D demonstration responses. One can also note that the classification never confuses 2D responses and VR responses.

Between the 3D responses and the 2D responses, 15 points were confused. Which is to say, responses that were identified as results from the 2D demonstration when they were results from the 3D demonstration, or as from the 3D demonstration when they were from the 2D demonstration. Whereas between the VR and 3D

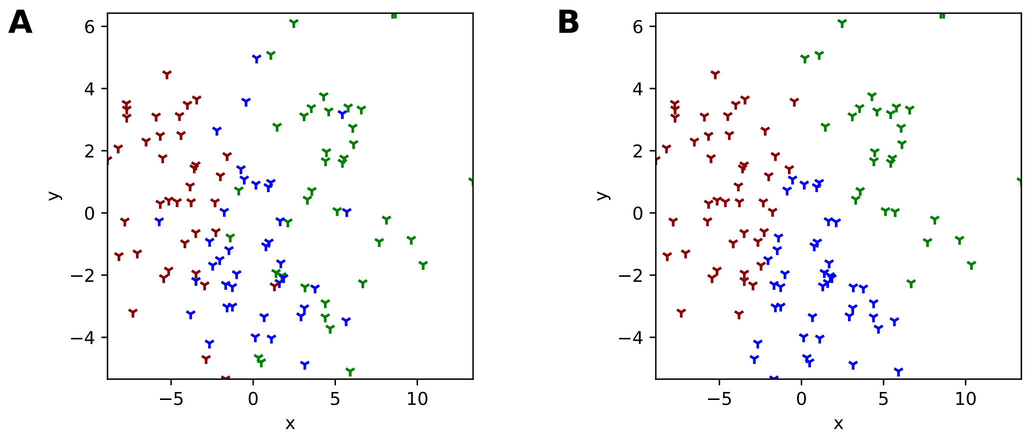


Figure 9.3: Plotting of first 2 principle components of the answers to the PQ coloured according to their (A) True labels (2D = Green, 3D = Blue, VR = Red) and (B) Ideal groups determined by k-means clustering for 3 clusters.

	2D	3D	VR
2D	26	3	0
3D	12	26	3
VR	0	9	35

Table 9.10: Confusion Matrix for Classification of Survey Responses

responses, only 12 points were confused in this manner. This analysis supports our earlier general conjecture that user responses suggest that the immersion and sense of presence in the three demonstrations was substantially different, and that the difference detected between the VR and 3D demonstrations was greater than the difference detected between the 3D and 2D demonstrations.

9.3 Summary

The quantitative analysis showed good results, with the data per frame reduction ratio of 99.98%, the results of the qualitative analysis seem to show that the system does not appear to have an equally substantial decrease in the quality.

A comparison to a similar system still leaves us with a per-frame data reduction of 99.75%. When offset by the data required to transmit our initial body scan, this reduction is lower. However, with a theoretical version of our system which periodically updates the body scan, at the shortest possible update interval, a reduction of 99.55% is achieved. This reduction was also achieved with a set-up time for processing and initial transmission of data which is considered reasonable, and a per-frame transmission time which is well within acceptable limits.

The qualitative analysis data seems to show a general pattern of the 3D demonstration obtaining better scores for user presence than the 2D demonstration, and of the VR demonstration obtaining better scores than the 3D demonstration. While other factors such as the novelty of virtual reality technology, the more detailed levels of environment in successive demonstrations and the fixed order of demonstrations could contribute to this, it can at least be determined that the body scan of the system is considered no worse than demonstrations in 2D and 3D environments of the same scan, which is an encouraging result, given the extra degree of scrutiny which was possible in the VR environment.

CHAPTER 10

CONCLUSION

Four main objectives were proposed for this study:

1. To build a system which reduced the amount of data needed to transmit a mesh animation across a network.
2. To evaluate the quality of the resultant system.
3. To determine the viability of the trade-off in quality required to make the system low-bandwidth.
4. To determine if this method of mesh animation transmission warrants use with better hardware and more complex software.

The quantitative analysis suggests the first objective was achieved, while the qualitative analysis suggest the second objective was also achieved. From these two objectives, it can be established that the third objective, and therefore the fourth objective, were also achieved.

A system was built which greatly reduces the amount of data necessary to send a body scan across a network, even in comparison to the reduction achieved by a system with similar aims. While the quality was reduced, it was determined that the reduction was not so great as to render the trade-off to be an unfavourable one.

10.1 Limitations of System

The system has several known limitations, the most notable of which are the restriction of its use for specifically human scans (which are identifiable as such by the Kinect sensor), the reduced quality of the resultant mesh which is animated and the permanent nature of the scan which is animated unless transmission is restarted. Each of these limitations are further discussed below.

10.1.1 Specificity

This system relies heavily on the scanning of what the Kinect sensor determines to be an actual human body, and as such does not extend well to applications outside of human body scans. Additionally, all scans for the testing of this system were performed on a person wearing clothing which does not excessively obscure the basic shape of the human body. Clothing which interferes with the reflection of infrared light or distorts the impression of the human body detected by the Kinect sensor will be unlikely to produce successful scans or animations for the system.

10.1.2 Quality

The scan produced by the system, with the arrangement of rigid cage meshes joined by linking meshes, does not produce the most natural looking animation of a human body. Additionally, no additional measures are taken to smooth any irregularities on the surface of the scanned mesh, or to normalize colours and heighten the quality of the texture obtained from the RGB camera of the Kinect sensor. This results in a mesh which is a reasonable representation of a human body and capable of being interpreted as such, but does not provide a high degree of realism in terms of surface, colour and animation.

10.1.3 Permanence

Once the body scan is transmitted, it remains constant in appearance for the course of the animation. A new scan can be created and manually retransmitted when a user wishes to reflect a change in the appearance of the person being scanned, but there is no functionality to address the updating of the mesh's appearance to reflect changes to the appearance of the person being scanned in real time.

10.2 Advantages of System

The system's primary advantages, over comparable systems, would be:

- Low Hardware Cost
- Simple Hardware Set-up
- High reduction of bandwidth for transmission of animation

The system also creates a body scan with a modular nature, which would allow for a more complex system to swap components of the scan with a variety of different components — much like the prefabricated components used for the hands and feet in this system.

10.3 Future Work

Future work on this system should likely focus on increasing addressing the limitations of the system and further utilizing advantageous aspects of the system.

The method used in this system prevents the use of a more general application to a variety of different mesh types. However, both the quality achieved by the system and the lack of mutability of the scan during use of the system can be addressed. The level of mesh detail achieved from the original scans, in geometric terms, is already quite high, although the limitations of the scanning technology create an imperfect visual representation of scanned objects. This can be addressed through various methods which were not implemented in this study, or by the use of superior scanning hardware. The permanence of the mesh — which is to say the inability of the mesh to update its appearance automatically while the system is in use — could be addressed by some functionality which automatically updates the scan or part of the scan during animation. This would produce a system which requires more bandwidth, but can deliver a system more directly comparable to other current body scan transmission methods.

10.3.1 Higher Quality Mesh and Animation Methods

Further work could be done to improve the quality of the scans used for animation in the system. Systems such as the one created by Li et al. [28] can produce high quality scans without additional set-up, and the system presented by Chen et al. [9] could provide high-quality textures to cover such scans. The implementation of the Laplacian mesh deformation system used by Li et al. [29] would provide greatly increased animation quality. Anthropometric scans to modify models, such as those presented in other studies [11, 51, 71, 17] could also be used to achieve higher quality meshes and animations, although sufficiently high quality models would need to be used to avoid bodies which feel generated rather than recorded.

10.3.2 Higher Quality Scans

The Kinect SDK for the current edition of the Kinect, has many useful tools and data streams which could be useful additions to this system, which could add to the realism and immersion of scans in virtual reality environments. The Kinect sensor's audio feed could be used in conjunction with the facial animation feed to create the abilities for the body scans to simulate conversation across a network. The Kinect Fusion utility could be used to fully scan a user environment to create a scene for these network applications. Multiple bodies could also be tracked at once, creating the possibility of single-camera multiple-user applications.

10.3.3 Mesh Updating

Currently, the animation phase on the computer receiving the scan mere passes through the skeleton animation received from the Kinect sensor over the network to the viewing computer.

The scanning computer could, however, be equipped with functionality which allows it to detect differences in the mesh surface being scanned using some differential image processing method, and send real-time or slightly delayed updates to the appearance of the mesh.

Alternatively, the system could be designed to deliver periodic mesh updates at fixed intervals, obviating the need for detection of differences in the mesh. This would keep the mesh automatically updated, for the cost of a higher bandwidth transmission. Such updates would, however, likely only update one side of the mesh (in a single Kinect system, such as the system currently is), require multiple Kinect sensors, or require a user initiated process where the user faces away from the Kinect sensor (as in the original scan) to obtain updated data for both sides.

10.3.4 Mesh Compression

Various forms of mesh compression, such as the method used by Mekuria et al., could also be used to compress the meshes transmitted across the system. [40] This would allow for a shorter transmission time of the initial body scan, and perhaps even allow for the continuous streaming of mesh components.

10.3.5 Mesh Components

A possible improvement, suggested by the advantages of the system, would be a use of the system in conjunction with a method which directly streams meshes to directly stream important component meshes — such as the face or hands — and replace objects from the original body scans with higher definition ongoing scans of these mesh components. This would achieve a realistic transmitted scan where facial expressions and gestures are clearly displayed, while the reductions of bandwidth made for the majority of the scanned body — which wouldn't need to be updated in nearly so much detail — would still be achieved.

10.4 Final Remarks

With the current competition in the market for VR devices, VR is likely to become a cheaper and more ubiquitous technology. Methods will need to be created to provide content for this platform, and while generated content is likely to be more prominent, we believe there is a need for content which comes from and more accurately reflects the real world, and that this content should be able to be created by as many people as possible.

This system does not currently deliver a visually perfect body scan, but the low requirements for bandwidth, processing power and cost are notable advantages. This system is a proof of concept for the method of animation of a mesh over a network with skeletal data achieves a substantial reduction in the amount of bandwidth required. With improvements to the quality, and used in conjunction with other methods to achieve a realistic and dynamic scan animation, this method would be very advantageous for use in the virtual reality telepresence systems.

Appendix A

System Code

The system code and a release file containing the complete Unity project will be available at:

`https://github.com/JujuZA/KinectBodyScanTransmitter`

The following system set-up to run the system is recommended:

- Windows 8.1
- Oculus Rift SDK version 0.6.0
- Unity 5.12
- Kinect SDK v2

Appendix B

Immersive Tendencies Questionnaire Results

This appendix contains the results of the Immersive Tendencies Questionnaire, which is an adaptation of one of the questionnaires created by Bob Witmer and Michael Singer to determine presence in virtual environments. [73]

These results contain answers to questions which establish the propensity of respondents to immerse themselves in tasks, entertainment media and their own thoughts. 38 participants were recruited to answer this questionnaire from the MIH Media Lab in the Engineering Faculty at the University of Stellenbosch, and the Information Science department in the Arts and Social Sciences faculty at the University of Stellenbosch. The exact questions used in this study can be found in Chapter 8.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	5	3	3	4	5	3	5	5	6	6	*	*	6	6	2	2	5	4	4	6	5	2	5	6	6	5	3	2	4
2	5	7	4	6	5	2	7	2	4	6	*	*	5	5	1	2	2	2	5	7	2	3	5	5	2	1	1	3	5
3	6	6	3	3	4	4	3	5	4	7	*	*	2	6	6	4	7	5	2	6	2	5	4	4	4	4	3	3	6
4	7	6	5	6	7	6	6	4	7	6	*	*	7	7	6	6	4	6	6	4	2	6	7	7	6	7	2	6	4
5	4	5	6	6	6	5	6	3	5	1	*	*	6	6	1	3	1	1	4	6	1	4	4	3	4	5	1	5	4
6	5	7	4	6	5	5	5	5	5	6	*	*	4	1	3	5	6	4	7	6	2	3	6	6	1	1	1	2	5
7	5	2	5	6	5	4	6	3	5	5	*	*	6	4	1	3	3	2	3	5	3	4	5	3	5	3	2	3	4
8	7	3	7	4	5	5	2	6	7	7	*	*	5	6	6	6	6	6	3	6	5	3	6	7	7	7	2	5	7
9	5	1	5	5	4	1	6	3	3	5	*	*	5	3	1	1	5	6	6	5	7	5	6	5	7	2	1	1	6
10	6	3	6	5	6	6	5	5	6	5	*	*	6	4	5	5	6	5	5	7	5	4	5	5	5	5	1	4	5
11	6	2	3	5	5	5	4	5	6	5	*	*	3	5	1	5	2	2	3	5	3	3	4	5	4	1	1	2	6
12	6	2	4	6	3	2	6	1	1	1	*	*	6	4	1	1	1	1	4	6	1	2	6	2	7	4	7	2	4
13	5	5	6	6	6	4	6	5	6	1	*	*	6	4	7	3	5	6	3	6	1	4	6	5	6	5	1	5	5
14	6	3	4	6	5	2	6	3	6	7	*	*	6	6	1	6	5	5	6	7	7	6	6	5	2	2	3	1	6
15	6	3	4	5	6	7	6	4	4	3	*	*	4	3	1	4	5	3	5	6	7	4	6	2	5	3	5	1	5
16	6	3	5	5	3	2	6	4	4	4	*	*	4	3	3	2	1	6	5	6	2	4	5	3	3	2	1	1	6
17	6	5	6	3	6	3	6	3	7	5	*	*	5	6	2	5	5	5	3	6	4	5	6	7	5	5	2	1	6
18	5	4	2	6	4	6	3	7	2	1	*	*	4	5	7	3	2	2	4	6	1	2	6	6	7	7	6	5	5
19	6	5	1	7	6	5	7	5	6	2	*	*	7	5	1	2	6	7	2	7	2	5	6	7	6	4	1	1	7
20	5	3	4	6	6	6	5	3	4	2	*	*	6	5	3	5	5	6	5	6	2	4	6	7	4	5	2	1	6
21	7	3	3	6	7	6	5	6	7	6	*	*	6	5	4	5	4	6	4	7	4	3	5	5	6	6	5	3	6
22	7	2	3	6	6	5	4	6	5	2	*	*	6	6	1	7	4	6	7	5	6	4	5	6	1	1	2	1	6
23	7	7	4	2	2	1	2	1	2	1	*	*	1	7	2	6	4	1	1	7	1	1	5	6	7	7	1	1	7
24	6	5	5	6	3	3	5	2	3	3	*	*	3	4	1	3	4	1	6	5	2	3	3	5	5	3	4	1	4
25	6	6	5	7	6	3	6	3	3	3	*	*	3	6	6	3	5	5	4	7	2	5	5	5	6	3	1	3	6
26	5	4	5	6	4	2	6	2	3	2	*	*	4	6	1	1	1	3	4	7	2	2	6	6	6	6	7	6	3
27	5	5	2	4	3	2	6	2	3	2	*	*	5	5	2	1	2	6	6	6	2	5	5	6	2	2	1	5	6
28	6	5	7	5	3	6	6	5	5	7	*	*	3	3	1	5	2	1	5	4	7	4	5	7	5	6	2	1	6
29	6	4	3	7	6	3	5	2	4	2	*	*	4	5	1	5	2	6	3	7	5	6	5	5	5	2	1	4	6
30	5	5	4	6	6	6	6	5	7	5	*	*	5	5	2	5	3	5	4	6	4	4	5	6	6	6	2	4	4
31	4	7	6	6	7	7	4	6	5	5	*	*	4	2	7	5	2	4	7	7	7	3	6	3	6	5	2	5	6
32	5	3	6	6	6	2	6	5	3	7	*	*	3	5	4	2	4	5	2	6	7	3	5	6	5	3	5	5	6
33	5	4	4	7	7	6	5	5	4	3	*	*	3	4	5	3	5	4	3	6	3	4	4	3	6	6	3	1	5
34	5	3	4	5	3	5	6	4	5	3	*	*	5	3	4	5	6	5	6	6	6	4	6	5	6	4	2	6	6
35	6	4	3	5	3	3	7	3	5	3	*	*	6	6	6	5	6	4	3	5	2	5	3	5	3	4	3	6	5
36	7	4	1	7	4	4	7	7	7	4	*	*	6	7	4	7	7	7	7	7	5	1	7	4	2	5	7	1	7
37	5	6	6	5	6	4	4	5	3	1	*	*	4	5	4	3	5	4	5	6	1	4	5	3	5	6	1	5	5
38	4	5	5	5	5	3	5	4	4	5	*	*	6	5	4	3	5	6	5	6	4	4	5	5	5	4	1	2	5

Table B.1: Immersive Tendencies Results for Questions 1-10 and 13-29.

*: Questions 11 and 12 of the ITQ did not follow the format of a seven-pointed scale. Question 11 was on a scale from 1 to 10, where 1 represented one or fewer books per month, and 10 represented 10 or more, and everything in between represented by the number chosen itself. Question 12 allowed a choice from a selection of popular genres, as well as the generic "Other Fiction" and "Other Non-Fiction" choices. The results of these questions are shown in Table B.2.

Participant	ITQ 11	ITQ 12
1	1	Fantasy
2	1	Adventure
3	2	Other non-fiction
4	1	Other non-fiction
5	7	Fantasy
6	1	Mysteries
7	1	Other fiction
8	6	Science Fiction
9	2	Historical
10	1	Other non-fiction
11	3	Science Fiction
12	2	Other non-fiction
13	2	Other non-fiction
14	2	Fantasy
15	2	Fantasy
16	1	Biographies
17	5	Fantasy
18	2	Autobiographies
19	1	Other non-fiction
20	1	Other non-fiction
21	1	Fantasy
22	1	Adventure
23	1	Other fiction
24	1	Other non-fiction
25	1	Autobiographies
26	2	Mysteries
27	2	Other non-fiction
28	2	Fantasy
29	1	Autobiographies
30	2	Fantasy
31	1	Autobiographies
32	4	Fantasy
33	1	Autobiographies
34	2	Adventure
35	3	Other non-fiction
36	1	Other non-fiction
37	1	Other fiction
38	2	Historical

Table B.2: Immersive Tendencies Results for Questions 11 and 12.

Appendix C

Presence Questionnaire Results

This appendix contains the results of the Presence Questionnaire for each demonstration conducted for this study, which is an adaptation of one of the questionnaires created by Bob Witmer and Michael Singer to determine presence in virtual environments.[73]

These answers show user impressions of their immersion and presence during the demonstrations conducted for evaluation of the system. The results for the 2D, 3D and VR demonstrations are shown in Tables C.1, C.2 and C.3, respectively. 38 participants were recruited to answer this questionnaire from the MIH Media Lab in the Engineering Faculty at the University of Stellenbosch, and the Information Science department in the Arts and Social Sciences faculty at the University of Stellenbosch. The exact questions used in this study can be found in Chapter 8.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	3	3	5	6	7	2	2	4	3	4	2	7	1	*	7	4	5	2	1
2	6	2	6	3	1	1	6	2	4	1	6	7	5	*	7	5	6	4	3
3	4	4	4	3	6	2	6	3	6	3	6	7	4	*	7	3	3	3	3
4	7	7	3	2	1	2	4	7	7	1	7	6	5	*	7	6	6	4	5
5	2	5	2	1	4	1	2	1	3	2	1	7	7	*	7	7	6	3	6
6	1	3	3	6	3	2	4	1	2	5	3	3	3	*	6	1	2	1	1
7	6	3	6	5	5	1	2	2	2	1	6	6	3	*	6	6	6	6	6
8	5	3	3	6	4	2	3	2	2	1	1	5	4	*	3	4	4	3	3
9	5	4	6	2	1	1	2	2	3	1	6	7	7	*	7	5	5	5	4
10	5	4	5	3	5	3	4	5	5	1	5	6	5	*	4	4	5	4	3
11	2	2	2	2	5	1	2	1	2	1	1	7	3	*	7	4	6	6	1
12	7	4	5	4	2	2	2	2	2	2	3	2	3	*	1	2	2	3	2
13	2	3	1	5	4	1	4	4	4	1	3	4	4	*	3	6	5	4	2
14	3	6	5	3	2	5	5	3	5	1	5	7	6	*	5	3	4	2	2
15	4	3	4	5	5	1	6	3	4	1	3	6	7	*	3	3	4	4	1
16	4	3	3	3	4	4	4	3	4	5	5	3	5	*	6	5	3	3	2
17	5	4	5	2	3	1	1	1	6	1	7	7	6	*	5	3	6	5	3
18	3	5	4	5	6	5	3	3	5	5	6	3	6	*	4	7	2	4	3
19	2	2	2	4	5	4	1	1	2	6	2	1	1	*	7	5	3	2	1
20	6	5	4	6	4	1	2	6	7	1	3	7	2	*	3	6	7	5	3
21	4	3	4	6	2	2	3	3	2	5	3	4	2	*	3	4	4	3	5
22	6	3	3	4	2	1	3	1	2	1	2	7	4	*	7	2	2	4	5
23	3	3	3	1	1	1	1	4	7	1	4	7	1	*	7	7	7	4	4
24	2	3	2	3	1	1	3	1	3	1	3	2	1	*	6	6	4	6	3
25	3	5	4	6	5	7	6	6	6	1	6	6	5	*	6	7	6	6	2
26	7	5	6	2	1	1	5	7	7	1	7	4	2	*	6	7	6	6	4
27	5	5	5	5	5	5	4	6	7	2	6	5	6	*	7	5	6	5	6
28	2	5	6	4	5	6	5	3	6	2	6	7	3	*	2	6	3	3	1
29	3	4	3	5	6	2	4	5	3	5	6	4	3	*	6	5	5	3	2
30	3	3	2	2	2	1	2	1	2	1	1	7	1	*	7	5	6	3	1
31	3	3	4	5	5	4	3	3	5	1	6	4	2	*	7	4	3	3	3
32	7	6	5	5	3	2	5	6	6	1	2	6	4	*	6	4	5	4	5
33	3	3	2	4	5	6	4	5	5	6	4	4	5	*	3	4	3	3	2
34	3	4	3	2	3	1	4	2	3	5	5	4	7	*	6	2	2	2	2
35	4	3	3	3	2	2	3	2	4	3	3	5	6	*	3	4	3	2	4
36	1	2	1	1	7	4	3	3	7	1	7	4	2	*	7	3	7	2	4
37	3	1	1	1	1	1	1	1	1	2	1	7	1	*	7	1	1	1	1
38	5	4	6	5	5	3	3	3	4	4	5	5	4	*	4	3	4	3	2

Table C.1: Presence Questionnaire Results for 2D Demonstration.

*: Question 14 of the Presence Questionnaire deals with navigation. (See Section 8.2.2.1 and Table 8.2.) Users were informed to give any answer for this question, as those answers would be disregarded.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	6	6	6	4	3	4	4	6	7	3	6	6	6	6	7	7	6	6	6
2	4	3	6	2	4	5	4	6	6	1	6	7	6	7	7	6	6	6	7
3	4	4	4	4	5	3	7	6	5	1	6	7	4	3	7	4	4	4	4
4	5	6	6	2	2	2	4	5	5	1	5	6	6	4	7	6	3	3	6
5	3	7	3	1	1	3	1	2	5	1	5	7	4	3	7	7	6	5	6
6	4	4	5	3	5	4	4	4	4	2	3	6	3	4	5	4	5	4	5
7	4	4	6	5	5	1	6	4	5	1	5	6	3	5	3	5	6	5	7
8	6	5	5	5	5	5	5	6	6	1	6	6	5	6	6	6	6	5	6
9	1	5	6	5	2	1	3	4	5	1	5	5	5	5	7	5	5	5	5
10	5	5	5	3	3	4	5	5	6	1	5	7	5	4	2	5	6	5	6
11	4	3	4	3	3	1	3	3	5	1	3	6	4	5	6	4	6	4	6
12	2	5	4	6	2	5	5	5	6	4	6	3	6	4	3	3	2	2	2
13	4	5	5	6	5	4	5	6	6	3	5	3	5	5	4	4	2	4	6
14	5	5	6	2	3	2	6	2	7	1	5	6	5	4	6	5	6	4	7
15	4	4	5	6	3	1	3	4	4	1	6	7	7	4	6	5	5	3	4
16	5	5	4	3	3	2	4	5	5	3	5	3	3	4	6	5	5	5	5
17	6	1	6	5	4	3	1	5	7	1	7	6	6	4	5	6	6	6	6
18	4	5	3	6	7	6	5	4	4	2	6	5	5	4	4	5	3	2	4
19	6	6	7	6	7	5	6	6	6	2	6	5	6	6	5	6	5	5	7
20	3	5	4	6	5	5	4	6	5	1	5	6	7	3	4	5	6	5	6
21	6	6	5	6	3	2	5	5	5	2	6	5	3	4	3	5	6	5	7
22	1	2	1	3	4	1	2	6	3	1	5	6	4	2	6	4	2	1	6
23	1	4	4	1	1	1	1	5	5	1	4	7	1	4	7	4	4	3	4
24	3	4	4	3	1	1	3	2	6	1	6	6	2	5	6	5	2	3	5
25	3	6	5	6	4	1	5	6	7	1	7	7	6	1	5	6	7	6	6
26	4	7	6	1	1	1	5	7	7	2	6	6	1	2	6	3	6	3	5
27	4	5	5	6	6	5	5	6	5	2	6	7	6	4	7	6	6	5	6
28	3	5	6	5	3	3	5	4	5	4	5	6	4	4	4	5	5	3	4
29	5	5	4	3	6	6	5	3	4	3	6	6	2	3	6	5	5	4	3
30	2	4	6	4	4	1	2	4	6	1	4	6	4	4	5	5	5	5	5
31	5	6	6	6	4	3	4	4	5	1	7	5	5	5	5	5	6	5	4
32	3	7	6	4	4	4	5	5	6	1	5	7	5	3	6	4	6	4	7
33	4	4	5	5	6	3	4	5	5	3	5	6	4	5	3	4	3	5	5
34	3	5	5	4	4	3	3	4	5	4	5	6	5	4	5	4	4	3	5
35	4	4	5	5	5	5	4	5	6	2	5	5	5	5	6	5	3	2	5
36	3	4	3	1	2	4	4	4	3	1	7	7	4	2	7	5	6	4	6
37	1	3	4	3	4	2	3	3	3	4	4	4	4	4	6	3	2	1	3
38	4	5	6	2	5	5	3	6	6	2	6	6	4	4	3	5	6	5	7

Table C.2: Presence Questionnaire Results for 3D Demonstration.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	6	7	7	2	5	7	1	7	7	7	7	6	4	5	3	7	6	3	7
2	2	1	7	7	6	6	5	7	7	1	7	7	7	5	5	5	2	6	6
3	6	7	6	1	1	5	3	7	7	5	7	7	6	4	5	5	5	5	5
4	6	7	7	6	7	7	6	7	7	6	7	6	6	4	2	6	4	4	7
5	1	3	3	6	2	5	5	7	7	7	7	2	4	4	2	6	5	6	7
6	7	7	7	7	5	7	6	6	7	7	7	4	7	2	1	4	7	6	7
7	3	5	6	6	5	5	5	6	6	3	6	5	6	4	3	5	5	5	6
8	7	6	7	6	7	7	7	7	7	1	7	7	6	4	2	7	7	7	7
9	5	5	6	6	5	5	5	6	6	2	6	6	6	5	6	5	5	5	6
10	3	7	7	2	3	5	4	6	7	6	7	4	2	3	2	6	6	5	7
11	4	6	6	6	6	5	5	7	6	2	6	4	5	5	3	6	6	5	7
12	5	6	6	7	6	4	2	6	5	2	7	6	2	4	4	4	4	5	6
13	6	7	7	6	5	6	6	6	7	5	6	6	6	6	2	5	3	3	7
14	6	7	7	6	5	6	5	7	6	2	7	6	3	2	3	6	3	6	6
15	5	5	5	2	2	2	2	6	6	2	5	6	5	4	6	4	3	2	6
16	2	5	5	6	3	6	5	5	6	2	6	5	4	5	4	3	4	5	6
17	6	1	7	4	6	6	1	7	7	2	7	6	6	5	3	6	5	5	7
18	7	6	5	7	4	2	6	7	7	2	6	6	2	3	3	7	7	4	6
19	7	6	7	7	6	1	7	7	7	1	7	7	7	7	2	7	7	6	7
20	6	6	7	4	2	6	6	6	6	6	7	5	6	2	5	6	3	3	6
21	7	7	7	6	7	6	4	7	7	6	7	3	6	6	2	7	7	7	7
22	2	7	7	7	6	5	6	6	6	5	7	5	7	6	2	4	3	3	5
23	7	5	7	1	1	5	3	6	7	2	7	7	1	4	7	7	5	4	7
24	5	6	6	6	2	6	5	7	7	7	6	3	4	4	4	6	3	6	5
25	5	7	7	7	5	7	6	7	7	3	7	6	5	5	3	7	7	6	7
26	7	7	6	3	1	1	6	7	7	2	7	6	1	2	2	7	7	5	7
27	6	7	7	6	6	6	6	6	7	1	7	7	7	1	6	6	6	6	7
28	4	7	7	7	3	6	5	7	6	6	7	5	5	2	2	5	5	6	7
29	6	7	7	3	6	7	6	7	6	5	7	4	3	4	3	6	5	5	5
30	4	6	5	6	4	6	5	6	6	6	6	4	6	3	3	6	4	3	7
31	6	7	7	7	5	3	7	7	7	2	7	5	7	5	3	7	6	6	7
32	6	7	7	2	3	7	7	7	7	2	7	6	4	5	4	5	5	4	7
33	6	6	7	7	4	5	5	6	7	6	6	3	3	3	4	5	5	5	6
34	3	6	5	7	3	4	4	6	6	4	6	4	3	3	4	5	5	4	5
35	5	6	5	7	6	6	5	7	6	7	7	3	7	4	6	6	6	5	7
36	6	6	3	6	1	5	5	6	7	1	7	6	5	3	3	4	3	3	7
37	5	7	7	6	6	7	6	7	7	6	7	5	6	6	3	6	6	5	7
38	6	7	7	7	7	7	5	7	7	6	6	3	5	5	3	6	6	6	6

Table C.3: Presence Questionnaire Results for VR Demonstration.

List of References

[1] Michael Abrash. What VR Could, Should, And Almost Certainly Will Be Within Two Years. *Valve Software, Steam Dev Days Presentation*, 2014.

[2] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *Vis. Comput.*, pages 105–114, 2003.

[3] Marc Alexa and Wolfgang Müller. Representing Animations by Principal Components. *Computer Graphics Forum*, 19(3):411–418, 2003.

[4] Mohammah Alizadeh. Realistic Hand 3D Model, 2015.

[5] Anton Artaud and Mary Caroline Richards. *The Theater and Its Double*. Evergreen book. Grove Press, 1958.

[6] Fausto Bernardini and Holly E. Rushmeier. The 3d model acquisition pipeline. *Comput. Graph. Forum*, 21:149–172, 2002.

[7] Damien Broderick. *The Judas Mandala*. Pocket Books, 1982.

[8] Tolga K. Capin, Joaquim Esmerado, and Daniel Thalmann. A dead-reckoning technique for streaming virtual human animation. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(3):411–414, 1999.

[9] Chih Fan Chen, Mark Bolas, and Evan Suma Rosenberg. Rapid creation of photorealistic virtual reality content with consumer depth cameras. In *Proceedings - IEEE Virtual Reality*, pages 473–474, mar 2017.

[10] Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Commun. ACM*, 35(6):64–72, jun 1992.

[11] Yan Cui, Will Chang, and Didier Stricker. Fully automatic body scanning and motion capture using two kinects. *SIGGRAPH Asia*, pages 1–1, 2013.

[12] Dante D'Orazio and Vlad Savov. Valve's VR headset is called the Vive and it's made by HTC, 2015.

[13] Benj Edwards. Unraveling The Enigma Of Nintendo's Virtual Boy, 20 Years Later, 2015.

[14] Morton L. Heilig. United States Patent 3,050,870, 1962.

[15] Iris Herbst, Anne-Kathrin Braun, Rod McCall, and Wolfgang Broll. Multi-dimensional Interactive City Exploration through Mixed Reality. In *2008 IEEE Virtual Reality Conference*, pages 259–260, mar 2008.

[16] Jerry L Hintze and Ray D Nelson. Violin Plots: A Box Plot-Density Trace Synergism. *American Statistical Association*, 52(2), 1998.

[17] David A. Hirshberg, Matthew Loper, Eric Rachlin, and Michael J. Black. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7577 LNCS(PART 6):242–255, 2012.

[18] Ken Horowitz. Sega VR: Great Idea or Wishful Thinking?, 2004.

[19] Mojtaba Hosseini and Nicolas D. Georganas. MPEG-4 BIFS streaming of large virtual environments and their animation on the web. *Proceedings of the seventh international conference on 3D Web technology*, pages 19–25, 2004.

[20] Dan Howdle and Mark Ashton. Worldwide broadband speed league 2018, 2018.

[21] John F Hughes, Andries van Dam, James D Foley, Morgan McGuire, Stephen K Feiner, and David F Sklar. *Computer Graphics: Principles and Practice*. The systems programming series. Addison-Wesley, 2014.

[22] ITU TCS. G.114: One-way transmission time, 2003.

[23] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics*, 24(3):399, 2005.

[24] Pileun Kim, Jingdao Chen, and Yong Cho. Slam-driven robotic mapping and registration of 3d point clouds. *Automation in Construction*, 89:38–48, 05 2018.

[25] Oliver Kreylos. Hacking the Kinect, 2015.

[26] Oliver Kreylos, Oliver G. Staadt, Dawn Y. Sumner, Gerald Bawden, Tony Bernardin, Magali I. Billen, Eric S. Cowgill, Ryan D. Gold, Bernd Hamann, Margarete Jadamec, and Louise H. Kellogg. Enabling scientific workflows in virtual reality. *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications - VRCIA '06*, (June):155, 2006.

[27] Claudia Kuster, Tiberiu Popa, Christopher Zach, Craig Gotsman, and Markus Gross. FreeCam: A hybrid camera system for interactive free-viewpoint video. *VMV 2011 - Vision, Modeling and Visualization*, pages 17–24, 2011.

[28] Hao Li, Jonathan T Barron, Gleb Gusev, and U C Berkeley. 3D Self-Portraits. *ACM Transactions on Graphics*, 32(6), 2013.

[29] Jituo Li, Guodong Lu, and Juntao Ye. Automatic skinning and animation of skeletal models. *Visual Computer*, 27(6-8):585–594, 2011.

[30] Jituo Li, Juntao Ye, Yangsheng Wang, Li Bai, and Guodong Lu. Fitting 3D garment models onto individual human models. *Computers and Graphics (Pergamon)*, 34(6):742–755, 2010.

[31] Jack M. Loomis. Distal Attribution and Presence. *Presence: Teleoperators and Virtual Environments*, 1(1):113–119, 2015.

[32] Milan Loviska, Otto Krause, Herman A Engelbrecht, Jason B Nel, Gregor Schiele, Alwyn Burger, Stephan Schmei\sser, Christopher Cichiwskyj, Lilian Calvet, Carsten Griwodz, and Pål Halvorsen. Immersed Gaming in Minecraft. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 32:1—-32:4, New York, NY, USA, 2016. ACM.

[33] Milan Loviska, Otto Krause, Jason B Nel, and Alwyn Burger. Puzzled. Processed. Meshed., 2016.

[34] Yao Lu, James Kwangjune Hahn, and Xiaoke Zhang. 3D Shape-based Body Composition Inference Model Using A Bayesian Network. *IEEE Journal of Biomedical and Health Informatics*, pages 1–1, 2019.

[35] Luciferbelzebuth. Cowboy boots 3D Model, 2014.

[36] James Macqueen. Some methods for classification and analysis of multivariate observations. *University of California Press*, 233(233):281–297, 1967.

[37] Sevastian Marevoy. Realistic Furniture And Interior Props Pack, 2018.

[38] Aman S Mathur. Low cost virtual reality for medical training. In *2015 IEEE Virtual Reality (VR)*, pages 345–346, mar 2015.

[39] Rufael Mekuria, Dimitrios Alexiadis, Petros Daras, and Pablo Cesar. Real-time encoding of live reconstructed mesh sequences for 3D tele-immersion. In *Electronic Proceedings of the 2013 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2013*, 2013.

[40] Rufael Mekuria, Pablo Cesar, Dick Bulterman, and Centrum Wiskunde. Low complexity connectivity driven dynamic geometry compression for 3D tele-immersion. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6162–6166, 2014.

[41] Rufael Mekuria, Antonella Frisiello, Marco Pasin, and Pablo Cesar. Network support for social 3-D immersive tele-presence with highly realistic natural and synthetic avatar users. *Proceedings of the 7th ACM International Workshop on Massively Multiuser Virtual Environments, MMVE 2015*, pages 19–24, 2015.

[42] Microsoft. Kinect API Overview, 2014.

[43] Microsoft. TCP-UDP, 2017.

[44] Microsoft. Binary Serialization, 2018.

[45] Suraj Raghuraman, Klara Nahrstedt, Karthik Venkatraman, Zhanyu Wang, Jian Wu, Jacob Clements, Reza Lotfian, Balakrishnan Prabhakaran, Xiaohu Guo, and Roozbeh Jafari. Immersive multiplayer tennis with microsoft kinect and body sensor networks. *ACM Multimedia*, page 1481, 2012.

[46] Juliana Restrepo and Helmuth Trefftz. Telepresence Support for Synchronous Distance Education Categories and Subject Descriptors. *ACM VRST*, pages 3–7, 2005.

[47] Adi Robertson. HTC's China-exclusive Vive Focus VR headset is now launching worldwide, 2018.

[48] Adi Robertson, Michael Zelenko, Katie Drummond, Casey Newton, and Melissa Smith. Voices from a virtual past, 2014.

[49] Peter Rubin. The Inside Story of Oculus Rift and How Virtual Reality Became Reality, 2014.

[50] Sam Rutherford. What the New Oculus Rift Offers Weary Gamers, 2019.

[51] Hyewon Seo, Frederic Cordier, and Nadia Magnenant-Thalmann. Synthesizing animatable body models with parameterized shape modifications. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 120–125, 2003.

[52] Ashish Shingade and Archana Ghotkar. Animation of 3D Human Model Using Markerless Motion Capture Applied To Sports. *International Journal of Computer Graphics & Animation*, 4(1):27–39, 2014.

[53] Sid Shuman. PlayStation VR: The Next Wave of Games Coming in Spring and Summer 2019, 2019.

[54] Mel Slater and CA Martin Usoh. An Experimental Exploration of Presence in Virtual Environments. *Department of Computer Science Technical Report No. 689*, pages 1–35, 1993.

[55] Mel Slater and Martin Usoh. Presence in immersive virtual environments. *Proceedings of IEEE Virtual Reality Annual International Symposium*, (c):90–96, 2002.

[56] Mel Slater, Martin Usoh, and Anthony Steed. Depth of presence in virtual environments. *Presence: Teleoperators and Virtual Environments*, 3(2):130–144, 1994.

[57] Aljosa Smolic, Ralf Sondershaus, Nikolce Stefanoski, Libor Váša, Karsten Müller, Jörn Ostermann, and Thomas Wiegand. *A Survey on Coding of Static and Dynamic 3D Meshes*, pages 239–311. 2007.

[58] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Mark Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. *Eurographics Symposium on Geometry Processing*, page 175, 2005.

[59] Aaron Souppouris. Sony's Project Morpheus is now 'PlayStation VR', 2015.

[60] Nick Statt. HTC announces new Vive Pro Eye virtual reality headset with native eye tracking, 2019.

[61] Neal Stephenson. *Snow Crash*. Bantam Books, 1992.

[62] Patrick Stotko, Stefan Krumpen, Matthias B. Hullin, Michael Weinmann, and Reinhard Klein. SLAMCast: Large-Scale, Real-Time 3D Reconstruction and Streaming for Immersive Multi-Client Live Telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1–1, may 2019.

[63] Ivan E Sutherland. A head-mounted three dimensional display. pages 5–10, 1968.

[64] Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan. Scanning 3D full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):643–650, 2012.

[65] Unity. Meshes, 2015.

[66] Unity. Oculus, 2017.

[67] Unity. Using the Network Manager, 2019.

[68] Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P. Brooks. Walking > walking-in-place > flying, in virtual environments. In *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 359–364, 2005.

[69] Martin Usoh, Ernest Catena, Sima Arman, and Mel Slater. Using presence questionnaires in reality. *Presence: Teleoperators and Virtual Environments*, 9(5):497–503, 2000.

[70] Thomas Waltemate, Dominik Gall, Daniel Roth, Mario Botsch, and Marc Erich Latoschik. The impact of avatar personalization and immersion on virtual body ownership, presence, and emotional response. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1643–1652, apr 2018.

[71] Qiming Wang and Sandy Ressler. Generation and manipulation of H-Anim CAESAR scan bodies. *Web3D*, page 191, 2007.

[72] Tom Warren. A closer look at HTC's new higher-resolution Vive Pro, 2018.

[73] Bob G. Witmer and Michael J. Singer. Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence: Teleoperators and Virtual Environments*, 7(3), 1998.

[74] Chao Zhang, Sergi Pujades, Michael Black, and Gerard Pons-Moll. Detailed, accurate, human shape estimation from clothed 3D scan sequences. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January, pages 5484–5493, jul 2017.