DEVELOPMENT OF A STANDARDIZED FRAMEWORK FOR COST-EFFECTIVE COMMUNICATION SYSTEM BASED ON 3D DATA STREAMING AND REAL-TIME 3D RECONSTRUCTION

A Thesis

by

DANG DUONG HAI HUYNH

BS, University of Science, 2014

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

May 2017

May 2017

DEVELOPMENT OF A STANDARDIZED FRAMEWORK FOR COST-EFFECTIVE COMMUNICATION SYSTEM BASED ON 3D DATA STREAMING AND REAL-TIME 3D RECONSTRUCTION

A Thesis

by

DANG DUONG HAI HUYNH

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Dr. Scott A. King
Chair

Dr. Ajay Katangur
Committee Member

Dr. Mohammed Belkhouche
Committee Member

May 2017

ABSTRACT

The common approachs for people to converse over a large geographical distance are via either SMS or video conference. A more immersive communication method over the internet that creates an experience which is closer to a face-to-face conversation is more desirable. The closest form is a conversation via the holographic projection of the participants and environment. Many motion pictures have featured this type of communication. While a complete system itself that uses holographic projection is still many years away, the core functions of such a system are not impossible to achieve now. Two such features include 3D reconstruction of the target and streaming of 3D data. With the current development speed of technology, 3D reconstruction can be achieved with cost-effective depth cameras and 3D streaming can be done after data optimization. The focus of this work is on how to approach the idea by using such devices to create a standardized platform for the implementation of the system with aforementioned features. Specifically, the system is able to capture 3D data from multiple depth sensors, reconstruct a 3D model of the human target to create an avatar, and stream the changes acquired from the sensors to the client to control the avatar in real time.

TABLE OF CONTENTS

CONTENTS PAGE

ABSTRACT v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

1 INTRODUCTION 1

1.1 Motivation 1
1.2 Contribution 2
1.3 Literature Review 2
1.3.1 Prior Work 2
1.3.1.1 Microsoft Holoportation 3
1.3.1.2 Oculus Avatar 4
1.3.1.3 VPARK 5
1.3.2 Background Information 5
1.3.2.1 3D Reconstruction in real time using depth camera . . . 6
1.3.2.2 3D Graphical Data Streaming 7

2 PROPOSED SYSTEM DESIGN 10

2.1 Problem Description 10
2.2 Proposed System Structure 11
2.3 3D Reconstruction in Real Time using Kinects 12
2.3.1 Depth Device Data Registration 13
2.3.1.1 Registration Methods 13
2.3.1.2 Iterative Closest Points Algorithms 14
2.3.1.3 KeyPoints Estimation 15
2.3.1.4 Depth Sensor Error Model and Manual Transformation Adjustment 15
2.3.1.5 Contour Coherence based Registration Method 16
2.3.2 3D Model Mesh Building 18
2.3.2.1 Delaunay Triangulation Algorithm 18

2.3.2.2 Unordered Point Cloud Triangulation and Normal Estimation . . . 19
2.3.2.3 Poisson Mesh Reconstruction . . . 19
2.4 3D Streaming in Real Time of Humanoid Object . . . 20
2.4.1 Humanoid 3D Model Rigging . . . 20
2.4.2 Skeleton Streaming . . . 21
2.4.2.1 Open Sound Control - OSC . . . 22
2.4.3 Animating Humanoid 3D Model using Skeletal Data . . . 22
2.4.3.1 Forward Kinematics . . . 23
2.4.3.2 Inverse Kinematics . . . 23
2.4.3.3 Avatar Puppeteering . . . 23
2.4.4 Facial Rigging . . . 24
2.4.5 openFrameworks . . . 25
2.4.6 Point Cloud Library . . . 25
2.4.7 Microsoft Kinect SDK . . . 25

3 EXPERIMENTAL SETUP AND TESTING RESULTS . . . 26

3.1 Experimental System Setup . . . 26
3.2 3D Reconstruction Result . . . 27
3.2.1 Data Registration Result . . . 28
3.2.2 Mesh Generation Result . . . 31
3.2.3 Skeletal Rigging Result . . . 31
3.3 3D Streaming Result . . . 32
3.3.1 3D Model Streaming . . . 34
3.3.2 Skeletal Data Streaming . . . 34
3.3.3 Character Animation . . . 35
3.4 Result Evaluation . . . 37
3.4.1 Data Registration from connected Kinects . . . 38
3.4.2 Mesh Generation from registered data . . . 39
3.4.3 3D Character Rigging from the created mesh . . . 39
3.4.4 Stream the created and rigged mesh to the client machine . . . 39
3.4.5 Stream the captured skeletal data to client . . . 40
3.4.6 Animate the character . . . 40
3.4.7 Render the output data on the client side . . . 41

4 CONCLUSION AND FUTURE WORK . . . 42

4.1 Future Work 42
4.1.1 Depth Data Registration 42
4.1.2 Skeletal Data Reading 42
4.1.3 User Recognition 43
4.1.4 Character Modular Approach 43
4.1.5 Additional Data Stream 43
4.2 Conclusion 44

REFERENCES 45

LIST OF FIGURES

FIGURES PAGE

1 Star Wars Holographic. 1

2 Microsoft Holoportation[14] 3

3 Oculus Avatar[23] 4

4 VPARK[20] 5

5 System Structure Proposal. 11

6 Registration with ICP 14

7 ICP with matching key points.[17] 16

8 Registration using contour coherence.[21] 18

9 Poisson Surface Reconstruction.[10] 19

10 Rigging with deformable model.[4] 21

11 Animate joints with forward kinematic. 24

12 Experimental Setup. 26

13 Reconstruction and Rigging Pipeline. 27

14 Unregistered Depth Data from 4 Kinects. 28

15 Initial Rotations. 29

16 Rigid Registration. 30

17 Non-Rigid Registration. 31

18 Mesh Generation Result. 32

19 Automatic Rigging by fitting SCAPE model into the generated 3D mesh. 33

20 Streaming and Rendering Pipeline. 33

21 Mesh Transferred and Rendered. 34

22 Skeletal Data Streaming. 35

23 OSC Message with Skeletal Data. 35

24 3D Model Skeletal Rig Structure. 36

25 Virtual Interaction. 37

CHAPTER 1

INTRODUCTION

1.1 Motivation

Virtual Reality (VR) and Augmented Reality (AR) devices have become more and more popular. However, fully designed 3D content dedicated for these devices currently do not exist, let alone data streamed from other sources. One of the motivations for this research is to produce a cost-effective framework for streaming the 3D data to these AR/VR devices.

The desired outcome of this paper is to create data for a novel type of virtual-reality-based or augmented-reality-based conversations which are similar to what people seen in science fiction motion pictures. People can converse with an experience closer to a real life conversation, disregarding any geographical barrier. Figure 1 demonstrates what a science-fiction movie envisions a holographic conversation.

Figure 1. Star Wars Holographic.

Image coutersy of starwars.com

1.2 Contribution

The contributions of this paper are:

- Implementation of a system that allows the streaming of 3D data - This system can take data captured by the depth cameras, the Microsoft Kinect, and stream the data over a network connection. The receiving device accepts the data and renders the 3D content on the screen. For devices which were specifically designed for AR/VR, this data can be projected in 3D space around the users in real time and significantly enhance the user experience.

- Development of a system that can do complete 360-degree 3D Reconstruction - Utilizing multiple depth cameras to get the entire view of the target instead of moving one single camera or spinning the model around. This feature is necessary for the immersive experience which is required to achieve deeper interaction. The participants can move around freely in a designated area and still be visible as a whole model to the other participants.

1.3 Literature Review

Literature review for this project is separated into two parts. First part is the introduction to some applications that have similar features. The second part is about the underlying techniques which are used for this work.

1.3.1 Prior Work

This section introduces some systems that have similar features or methodologies of this paper. This also includes the caveats of each implementation which leads to the

approach which is utilized in this project.

1.3.1.1 Microsoft Holoportation

Holoportation, which was introduced by Microsoft[14] can create a 360-degree view of a humanoid 3D model. Holoportation features the expected outcome of this project. The system requires a high-end workstation with two powerful GPUs. The two GPUs are programmed to work in a pipeline fashion to process and stream data from specialized depth cameras. While this setup has high accuracy and speed, the cost to set up and maintain such a system is very high and not consumer friendly. Our work is more focused on how to make the system's price and setup more cost-effective based on the consumer-grade depth sensors. Figure 2 shows Microsoft Holoportation's data being viewed through the Hololens.

Figure 2. Microsoft Holoportation[14]

1.3.1.2 Oculus Avatar

Oculus[23] introduced Oculus Avatar for the users to manually create a digitized version of themselves in the virtual world. This implementation features high detail of the users hand movements and facial expression. To achieve that, this setup requires specialized equipments which are attached to the user body. While the capture of user action is precise, it is limited to the hands and the head. The attachments could also lead to discomfort for user after extended usage period.

This application is introduced by Oculus[23] to manually create a digitized version of themselves and control these avatars inside the virtual world. This main focus of this implementation is the detail of the hands' movements and facial expression. This type of setup, consequently, limited to the hands and the head area. This will make the avatars can only look like a bust statue of a person. Oculus Avatar also requires special equipment attached to the users' body, which can cause discomfort over time.

Figure 3. Oculus Avatar[23]

1.3.1.3 VPARK

One of the older applications is called Virtual Park or VPARK that was introduced by Seo et al.[20]. This implementation mainly focused on the creation of a virtual world, in which there would be avatars with realistic movements and speech. While the strength of this is the overall management of the virtual world and its actors, the downside is the standard the system is based on, which are MPEG-4 SNHC and VRML. Both have become obsolete and hard for anyone to maintain and develop as a result.

Figure 4. VPARK[20]

1.3.2 Background Information

The background informations in this section are separated into two main parts based on the knowledge that can be used to work toward fulfilling the requirements to create the features for the previously noted system. The two features are: 3D Reconstruction in real time using depth camera and 3D Graphical Data Streaming

1.3.2.1 3D Reconstruction in real time using depth camera

3D Reconstruction has been an active area of research in the field of Computer Science. The reason is that 3D Reconstruction can be applied to a vast range of areas in the industry such as Computer Graphics, Computer Vision, and Medical Imaging. Furthermore, 3D Reconstruction is also one of the most challenging subjects of research due to the demanding requirements such as the precision of the reconstructed model and the speed of the reconstruction process. Both of which ultimately lead to a need for such a requirement to be able to do the 3D reconstruction in real time.

At the beginning of this century, all of the listed demands of the 3D reconstruction process were not able to be met because of the limitation of hardware at the time. The price requirement for such a system also was not inexpensive enough to allow a cost-effective solution for the mass. However, since 2010, both hardware and software solutions for 3D Reconstruction have been gradually released and improved. Currently, some of the most popular hardware devices which are utilized for 3D Reconstruction are Microsoft Kinect, Google Tango, and Intel RealSense. All of these devices are based on the utilization of low-cost depth cameras to get the depth data (point cloud) and create the mesh based on these point clouds. The software for building the meshes from depth maps includes Kinect Fusion[8], Project Tango SDK[5], and Intel RealSense SDK[6]. These software solutions usually utilize existing algorithms such as Iterative Closest Point (ICP)[16], K-Means, etc. These algorithms are applied to the two main steps to generate 3D meshes from the point cloud. The first step is usually to separate the foreground, which contains the point clouds that make up the desired model from the background. The second step is to generate triangles from the points by some of the algorithms which were studied thoroughly

and proved to be effective. As a result, it is possible to do the 3D reconstruction in real time.

1.3.2.2 3D Graphical Data Streaming

Currently, there has been no widely used format for 3D data transfer over a network in real time in the industry. Some formats which were dedicated for 3D data had already existed, such as Virtual Reality Markup Language (VRML) and MPEG-4 SHNC. VRML was designed to describe 3D image sequences and allow users to interact with them. MPEG-4 SHNC is based on the MPEG-4 standard, which is a method of compressing audio and visual for streaming. MPEG-4 SHNC provides compression scheme for VRML data. However, MPEG-4 has become obsolete and VRML suffers from poor efficiency and navigation in the scene. The most popular formats for streaming are either sound or image or the combinations of both, which are movies. Some existing papers have proposed the methods to stream the 3D data, which could be either point cloud or meshes. According to Maamar et al.[11], there are four main types of techniques when it comes to 3D streaming, which are Geometry Replication[9], Progressive mesh[7], Image-Based Rendering[2], and impostors[9]. Each of these techniques has its advantages and limitations during implementation.

- Geometry Replication[9] This technique is not very different from how modern computer games render 3D objects, which is to let the client store or download a copy of the 3D models geometry and generate the acquired meshes locally by the clients hardware. The advantage of this technique is the high level of scalability because every object is stored and handled separately. The main disadvantage would be the storage requirement of client to store the model.

- Progressive Meshes[7] Similar to how the browsers download and display a large image file, this technique involves rendering and transmitting object progressively. The model will be separated into a low-poly and a high-poly version. The low-poly that is smaller in size will be sent first to the client. Then it will be updated over time to reach the original quality. This way, the model will be rendered faster than waiting for the high quality model to be transfered. The drawback is that the time required for the complete mesh generation would increase.

- Impostors[9] As the name implies, in the impostor technique, the data which is going to be streamed is not the actual 3D data. What is being streamed is an image that is converted from the 3D model by the server. Because the 3D meshes are not going to be transferred, but the conventional image data is, this technique can take advantage of the current streaming technologies for 2D rasterized data. The rendering time and bandwidth requirement also can be lowered as a result. The reason is that the client now only needs to map the received image to a shape as texture. However, it can cause the lack of image depth to objects which are close to the users.

- Image-Based Rendering[2]: Taking the impostor technique one step further, data being streamed in image rendering technique is not only the converted image data from the 3D meshes but also the data that would represent the models instead of the geometry itself. This allows further enhancement of streaming due to the need of the client to render the scene. What also happens is the problem with displaying such as distortion or pixel loss. An application that suffers from the same weakness is the panorama photograph recoring application. More often than not, the stitching of different photo frames into one giant picture usually results in the images are not

lined up corrently.

The next part of this paper is about the actual research done in order to complete the mentioned objectives, which are the 3D reconstruction and 3D data streaming. The final product of the research is a system which is capable of performing both of these functions, which includes the reconstruction of the humanoid target and stream the avatar, along with its changes to the client to emulate the virtual interactions.

CHAPTER 2

PROPOSED SYSTEM DESIGN

2.1 Problem Description

Even though when it comes to 3D Reconstruction, there are already many well-studied algorithms existed along with appropriately designed implementations. However, most of the current implementations are limited by the camera perspective, which can only capture a partial view of the desired object and completely unable to see what is outside of its viewport. Most 3D Reconstruction or 3D Scanner right now require either the camera to be moved around the object or the object to be spun itself so that a complete model can be captured. Such workaround cannot work for the goal of the system described in this research due to a need of a much higher level of immersion.

For the streaming portion, compared to the conventional methods to stream other popular types of data such as sound or video, 3D streaming shares some of the challenges when streaming those data, such as:

- Wireless network bandwidth limitation: Due to the amount of streamed data being larger, all the drawbacks of the wireless network have more significant effect on the 3D streaming. Some of these wireless network related problems are background noise, interferences, etc.
- Streaming performance: Similar to the issue of the wireless network bandwidth limitation, but affected by the overall state of the global network infrastructure.
- Density: Streaming a node with high density can cause high communication overhead, which results in packet collision while a low-density node may cause the packet to be dropped. This can cause a large impact to 3D streaming because of the natural

disparity in the density of 3D data.

Apart from the similar challenges while streaming 3D compared to streaming other conventional data types, 3D streaming has its distinctive problem, which is:

- Client performance limitation: For 2D streaming, most of the current generations clients have sufficient power to render the content and many efficient algorithms are used to support the streaming. But when it comes to 3D Streaming, the requirement of performance for the client side is significantly higher. Furthermore, the performance also varies from device to device, which makes it harder to decide on a standard quality of 3D data. In other words, it is a challenge to choose a level of detail of 3D model so that most of the current generation devices can handle without compromising the experience.

2.2 Proposed System Structure

To solve the described problems, a 3D Streaming System is put together like Figure 5

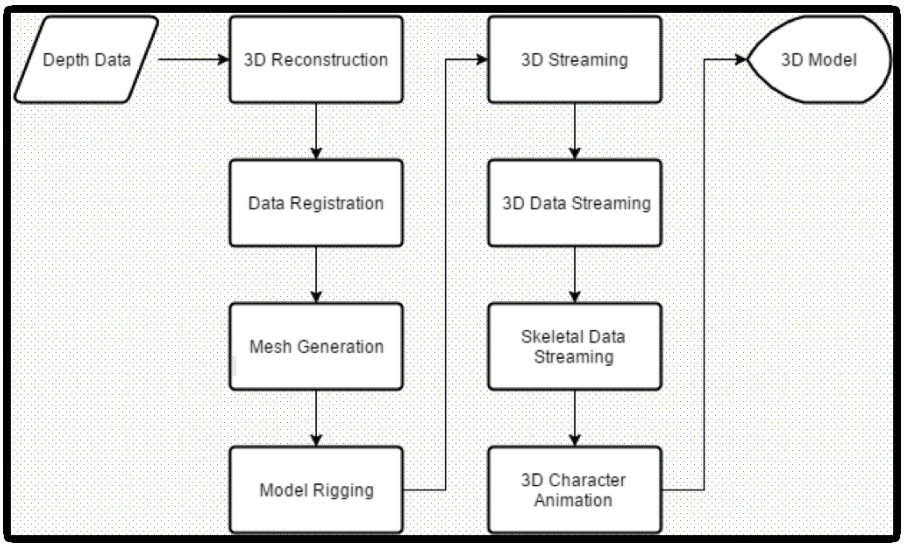

Figure 5. System Structure Proposal.

Multiple Kinect Devices (4 sensors are used for this setup) are connected to one single computer for data acquisition. Theoretically, only one Kinect is supported per USB Controller. To add more sensors to the system, more USB controllers need to be installed via PCI. The program is written in C++ based on the openFrameworks toolkit. This program provides ablilites to calibrate the input data from each Kinect by performing data registration process.

Calibrated data from all sensors will be processed by a down-sampling algorithm, then triangulation and normal estimation to acquire a complete 3D model. This completed 3D model will be rigged and sent to the client once.

The system will track the user skeletal information with the Kinects and control the pre-defined model with this information. The skeletal information is also streamed to the socket client, which will do the same thing and display to the other users of the system. The method for controlling the 3D model is based on Inverse and Forward Kinematics.

2.3 3D Reconstruction in Real Time using Kinects

To implement a 3D Reconstruction System based on the Kinects, at least 3 or more devices should be used for capturing data. The reason is because the limitation of the Kinect camera's frustum is that it only allows a maximum of 120 degree view of the desired object. In other word, in case of the humanoid object of this research, one Kinect can only provide approximately 40 percent of of the human body. That is why it is safe to assume that a minimum of three Kinects can provide a 360 degree of the target's body. With this as the starting point, next problems can be addressed as followed:

2.3.1 Depth Device Data Registration

Every camera has it's own coordinate system. To actually combine data from multiple devices into one single system, it is mandatory to find out the extrinsic of each device. The extrinsic of each Kinect device consist of the transformation matrix that will rigidly transform all data from one device to another. After doing the extrinsic calibrations to determine the extrinsic parameters, the original data of each device's coordinate system will be move into the same predefined "world space". This calibration process is usually called Extrinsics Calibration or Point Cloud Registration in case of depth sensor devices.

2.3.1.1 Registration Methods

There are many ways to find the relative location of a camera in space. For the normal RGB camera, this can be done by implementing the algorithm to solve the Perspective-n-Point problem. If two or more cameras [24] are set to look at one calibrating pattern and capture the corresponding 2D points of the pattern, the calculated matrices will take the cameras to the coordinates of the calibrating pattern. Unfortunately, these algorithms require intrinsics calibration for each cameras and heavily relied on the quality of the images or specifically the camera sensor's quality. Therefore, they will not work well with the depth sensors such as Kinects due to these devices are equipped with a somewhat low quality RGB sensor. The Kinect 360's RGB sensor maximum resolution is 1280x1024 at low frame rate. The default RGB stream with 30 frames per second can only output VGA resolution (640x480).

However, since the devices in use are depth sensors, the registration can be done directly to the returned data from the devices, the point clouds. There are also many algorithms available to perform the point cloud registration. The most popular one

is Iterative Closest Point, which tries to reduce the difference between two sets of point cloud.

Another problem specific to this research is the use of multiple Kinects. Since the devices are set up so that they will be capturing data from different angles, the resulting depth data from each devices will be very different from each other. That means the overlapping data, or the corresponding points in both set of point clouds will not be high, which wil result in the quality of the ICP algorithm will be affected. To increase the performace of the ICP algorithm, another algorithm is implemented to find the correspondence matching keypoints of both point clouds. The ICP will only work on these keypoints so that it will improve the algorithms performance.

2.3.1.2 Iterative Closest Points Algorithms

With the estimated correspondences, the last step of the registration process is to compute the rigid transformations using the remaining good correspondences using Iterative Closest Points Algorithms. The final results will be the transformation matrix that will take all data from one Kinect to another. The process is repeated until all devices are calibrated. Figure 6 demonstrates the implementation of ICP to register two set of point clouds.

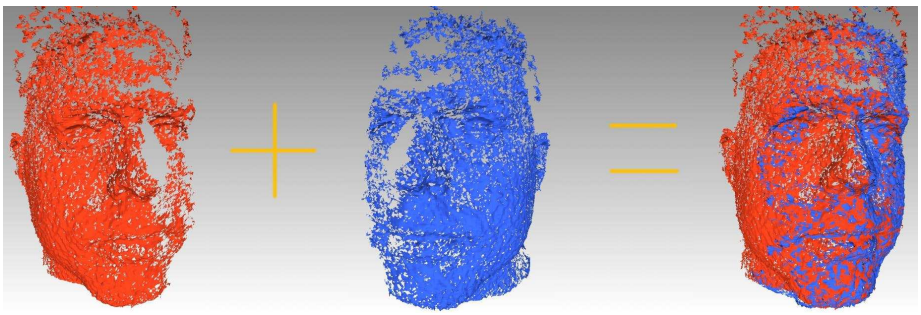

Figure 6. Registration with ICP

2.3.1.3 KeyPoints Estimation

The "KeyPoint" is a speficic point that carry an "interestingness" [17] , which can be extracted and used as the feature descriptor of the scene. There are various types of Keypoint that can be implemented such as Scale-invariant Feature Transform (SIFT), Normal Aligned Radial Feature (NARF), and Features from Accelerated Segment Test (FAST).

After extraction, the Keypoints will be compared to summarize some characteristics of the Keypoints in a vector format. These feature discriptors should be independent of keypoint position, consistent result against the image transformations and also have to be scale independent.

Using the resulting two sets of feature descriptors coming from 2 Kinects, Point Matching of Feature Matching can be applied to find the Correspondences in both scenes.

Obviously, not every computed correspondences are accurate. The reason is the set up of multiple devices will only provide a partial overlap of the scene. The correct correspondences will only be in the overlapping parts. Therefore, the incorrect estimate should be rejected using RANSAC or manually remove the non overlapping data from each scene. Figure 7 shows the ultilization of ICP only with found key points.

2.3.1.4 Depth Sensor Error Model and Manual Transformation Adjustment

One limitation of the depth sensor such as Kinect is the accuracy of the returned depth data not being consistent. For instance, the optimal range for the Kinect is around 3 meters, within certain range of that distance, the precision is high. However, the further away from the camera, the less accurate the point cloud's point value

is going to be. That's why it is necessary for human supervising. In other word, it requires manual transformation to increase the rigid transformation precision for each Kinect.

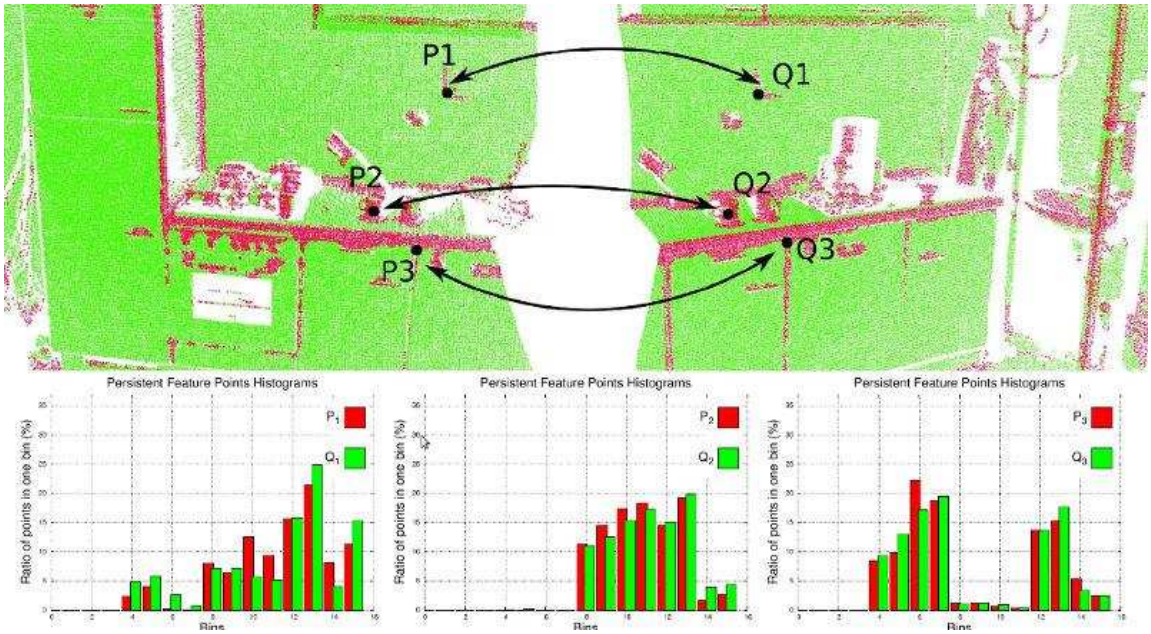

Figure 7. ICP with matching key points.[17]

2.3.1.5 Contour Coherence based Registration Method

There is another problem with using the registration method based on ICP algorithm. The algorithm works best when the two sets of input data fulfill both of the following requirements, both set of the input datasets must have a high level of overlapping data and the actual spatial difference between two frames has to be as little as possible. For application such as Kinect Fusion, the reason that ICP works really well when the device is moved slowly across the scene is because both of these requirements are met. When the Kinect is moved fast enough, the ICP algorithm starts failing to minimize the distance between frames because the difference is too

great.

In this work, we use multiple Kinects which are setup in a stationary manner that look at one particular location. To avoid cross-talking, only one Kinect is positioned at 90 degree from each other, which results in total four Kinects being utilized. The reason that four Kinect is the optimal number is:

- Lower cross-talking: more Kinects looking at the same view will suffer from infrared interference. One Kinect at every 90 degree will still have enough data overlap without causing too much interference.

- Lower USB Bandwidth: a typical personal computer features two main USB 2.0 controllers, the front and the back. Each USB 2.0 controller can only carry two Kinect devices. More Kinects connected to the system will required adding custom USB controllers.

The four-Kinect setup will return the difference between each frame coming from every Kinect approximately 90 degree of rotation for every two neighbor Kinects. Additionally, the portion of the targeted model would similarly be divided into four quarters, which put the overlapping portion for the frame-pair at a very low level. The combination of both of these conditions is unable to meet both of the requirement for ICP. As a result, the orthodox ICP algorithm approach will assumably not work in this case.

To solve this problem, Hao Li et al.[21] proposed using the registration method based on finding and minimizing the contour coherence between the two frames. This method works based on finding the same contour in the frame-pair and applying the Levenberg-Marquadt algorithm to solve the minimization. However, this only return a rough estimation of the Kinect pose that will register the body as a whole. Due to depth sensor error model which has been mention before, the misalign still exists for

smaller body parts where the errors are different between each frame. This can be solved by segmentating the body into smaller regions and eapply the contour-based registration on each part.

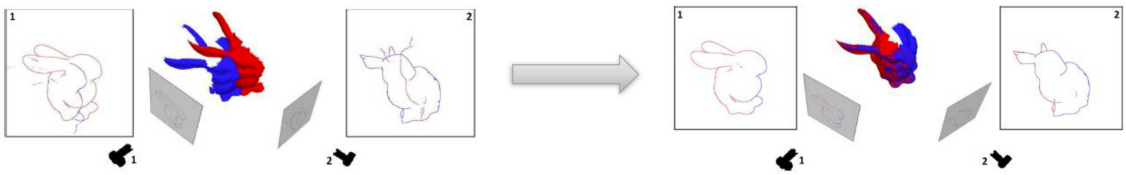

Figure 8. Registration using contour coherence.[21]

2.3.2 3D Model Mesh Building

After the calibration process, all data coming from every sensor are now in the same coordinate space. The next step is to generate the 3D mesh out of these points [3] . However, the problem is that all the points in the cloud are most likely unordered. To make it possible for these points to become polygons of the 3D model, these points will require triangulation and normal estimation to be rendered.

2.3.2.1 Delaunay Triangulation Algorithm

The Delaunay Triangulation Algorithm provides a way to triangulate a set of points which are not on a same line on a plane. The basic idea of Delaunay Triangulation is to find the set of indices of points that make up a triangle such that there are no points other than the three exist inside the circumcicle of that triangle. However, this can also be expanded for n-dimensional dataset such as point cloud that consist 3-dimensional data. In the case of point cloud, instead of using the circumcircle in 2-dimensional case, the circumscribed spheres of the triangles are considered.

2.3.2.2 Unordered Point Cloud Triangulation and Normal Estimation

The resulted registered point cloud data is a perfect example for large and noisy data. A method proposed by Marton et al [12] can take care of performing fast surface reconstruction from large and noisy dataset such as the one being used. This method can create the geometrical surface of the object based on data resampling and reliable triangulation algorithm in almost real time.

2.3.2.3 Poisson Mesh Reconstruction

Another method to reconstruct the 3D model is to consider the process as a spatial Poisson calculation. According to Kazhdan et al. [10], the Poisson formulation can be applied to the surface reconstruction problem. Since the equation considers all the points at the same time, the result is that data noise becomes a minor problem. Combining with the returned data from the Kinect that also includes color for each point, the texture of the model can be the direct color of each vertex.

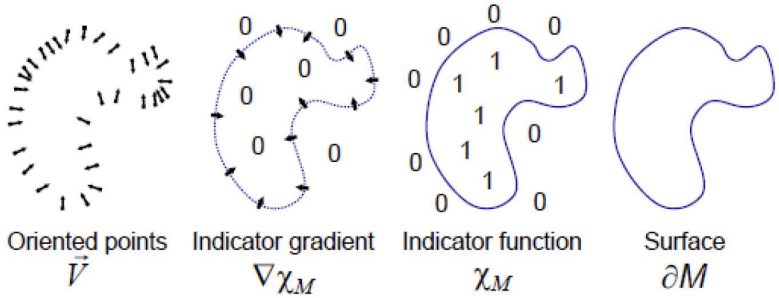

Figure 9. Poisson Surface Reconstruction.[10]

2.4 3D Streaming in Real Time of Humanoid Object

After creating a 3D model out of the raw data from the depth devices, instead of streaming the whole 3D model over and over again, the system rather streams the changes of the model rather than the whole mesh. Specifically, since we are targeting the streaming of changes in humanoid character, the most viable data to be streamed would be skeletal data. The skeletal joint data can then be used to control the humanoid model [22] . This would be similar to motion capture systems, where targeted joint locations are tracked and used to move a generated 3D model. This method requires the skeletal information and the humanoid 3D model to be rigged with joints. The skeletal data can be acquired from the Kinects thanks to the built in trained dataset. What is left is to rig the newly created 3D model and stream the skeletal data for control

2.4.1 Humanoid 3D Model Rigging

For a humanoid 3D model, a rig is basically the digital skeleton which is set to the mesh. The rig also contains joints and bones, which will be used as handles to move the model to desired pose. In this case, the rig's joints will be controlled by the skeletal value from the Kinect. Given a humanoid 3D model, basic rigging involves these concept: Skeleton Placing, Joint Hierarchy, Degree of Freedom, and Facial Rigging. Normally, to create a high-quality skeleton rig, the rigging need to be done manually by aligning the skeletal structure to the 3D model, however, this process can be done automatically.

One of first automatic rigging process proposed by Baran et al.[1] is called Pinocchio. This algorithm works based on the process called skeleton embedding. The process calculate the location of the joints within the 3D character model by minimizing a

penalty function. While the algorithm works effectively and fast, it requires the 3D model to be almost perfect, specifically it has to be tight mesh with no holes. This is a potential problem for the 3D model generated from the Kinect because of the data noise and gaps. Most other methods after Pinocchio complement the original approach, which all start the process from the empty mesh to generate the skeleton rig.

Another different method, according to Feng et al.[4], is to start from an already generated set of skeleton template and transfer the rig to the empty mesh. The requirement is to have both of the pre-defined rig and the computing 3D mesh to be at the same location in the object space and have minimized difference in shape. Since the pre-defined template has high quality rigging information that just needed to be copy to the generated model, this method returns better results than Pinocchio approach and will be used for this project.

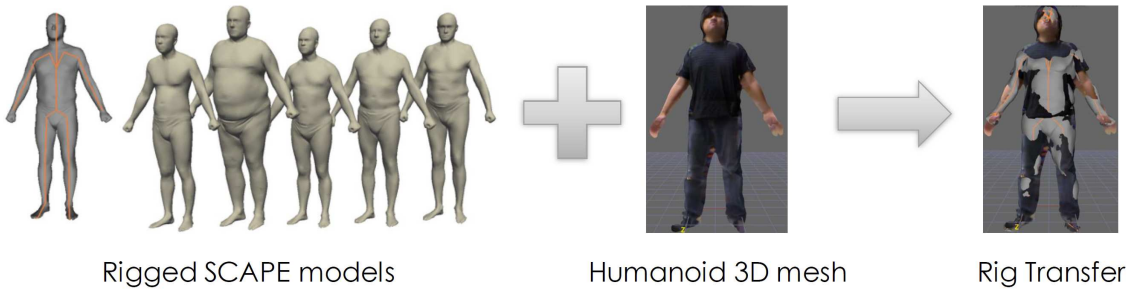

Figure 10. Rigging with deformable model.[4]

2.4.2 Skeleton Streaming

A Skeleton set provided by the Kinect can be streamed over the network. At the beginning of the stream, the system server provides the client with 3D mesh. After that, since the target won't have any substantial change over the specific streaming

period, what changes is only the rigid and non-rigid movement. Therefore, the skeletal data will be sent over to the client to control the 3D mesh. Because this needs to be in real time, the Open Sound Control, which has UDP protocol lying underneath is a good choice for streaming protocol.

2.4.2.1 Open Sound Control - OSC

Similar to XML or JSON, OSC is also a content format originally designed to share music performance data between musical instruments, computers, and other media devices. OSC offers interoperability, accuracy, and flexibility [19] . Using OSC messages to transfer the skeletal data from the Kinect allow the output data to be used in any language or framework of choice. An OSC recorder can also be attached so that the data can be saved and viewed later.

2.4.3 Animating Humanoid 3D Model using Skeletal Data

After acquiring a rigged 3D model and the skeletal data being sent over via OSC messages, the joints data will be used to control the rigged model. This section covers the rest of the Humanoid 3D model Rigging problem, which involves Forward Kinematics and Inverse Kinematics [13] . Depending on the animation properties, each scheme will be applied accordingly. Essentially, kinematics represents the study in figuring out the consequential state of a system based on the given parameters of the very same system while disregarding other aspects such mass. For instance, in mechanical engineering, kinematics is used to describe the motion of system comprise of smaller nodes which are linked hierarchially such as the human body skeletal joints.

2.4.3.1 Forward Kinematics

This is the direct way of animating the skeletal data. The joints with lower hierarchical status within the skeleton tree will follow the joints at higher lever. Basically, Forware Kinematics is used when the objective is to find out the target's next state, or the transformation to the next coordinate given the input angle. Because the children nodes inherit the transformation of their parents, this scheme is fairly forgiving and direct. Therefore this can be considered the "dumb" way of animation.

2.4.3.2 Inverse Kinematics

In this scheme, the child joints will have to adjust themselves to accommodate the Inverse Kinematics Joint. Contrary to the Forward Kinematics, Inverse Kinematics tries to figure out the needed angle to rotate the part to the location known before hand. In this scheme, the parents position and orientation are also determined by the position and orientation of the child nodes. As a result, Inverse Kinematics can be applied to create more complex motions more easily than Forward Kinematics. However, it also demands more thought put into the way the nodes are linked. This scheme can be called the "smart" way.

2.4.3.3 Avatar Puppeteering

The act of manipulating the avatar using the skeletal information can be called avatar puppeteering or avateering. Because the skeletal information comes from a pre-defined rigging template, all the joint data are already known and only require retargeting once to work with the joint data from the Kinect. For the Kinect One, the skeleton rig consists of 25 joints. These joints are connected in a hierarchial manner starting from the root, which is the pelvis in this case.

A joint is not just a location in space, it also has a rotation which is usually relative to its dirent parent. A bone is connected line segment between two joints. The joint rotation's basis is defined by: Bone Direction(Y axis), Normal(Z axis), and Binormal(X axis). Bone direction always match the skeleton. Normal is the joint roll, always perpendicular to the bone. Binormal is cross product of the normal and bone direction. To apply the animations to the character, the absolute location of the root will be applied to the root, then apply the relative rotation of each joints down the hierarchial tree. To simplify the process, usually the rotation will be against a default reference pose of the character.

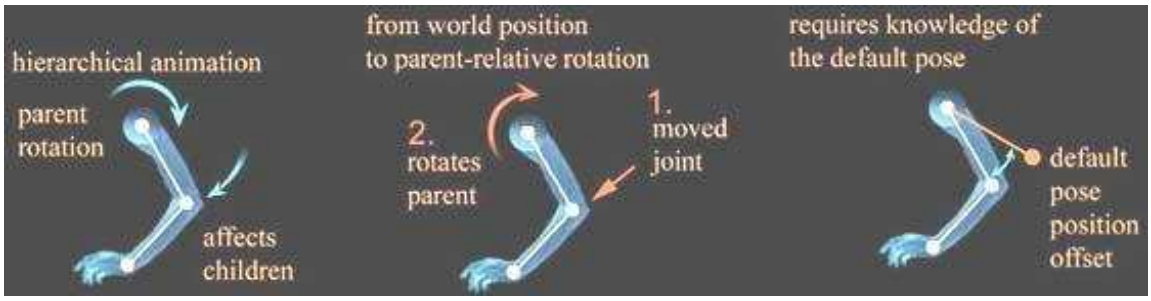

Figure 11. Animate joints with forward kinematic.

2.4.4 Facial Rigging

The final piece of the system is to match the facial pattern of the created 3D model to the captured target. This can be done by tracking the human face in the viewport of each Kinect of the system. The resulting data should be the tracked points of the human face laid out in the virtual space. After that, these points will be compared against the points on the face of the 3D model and transformed accordingly. With voice also recorded by the microphone arrays on each Kinect also being streamed, the final model will be able to move, performing facial expression and voice according

to the capturing target.

2.4.5 openFrameworks

openFrameworks is an open source toolkit for "creative coding". openFrameworks uses C++ as the programming language and has OpenGL running as its backbone [15] . It runs on almost all currently popular platform such as Windows, OS X, Linux, Android, and iOS. The main advantages of this framework are its simplicity, intuitivity and extensibility. This is the main framework to create the system's UI, Graphic Rendering, and Network Communication.

2.4.6 Point Cloud Library

Point Cloud Library (PCL) is a standalong, large scale, open source library for 2D/3D image and point cloud processing. This library contains most of the state-of-the-art algorithms for working with 2D or 3D data [18] . Similar to openFrameworks, this library is also cross-platform and based on C++ language. The PCL will be mainly be used for processing the input depth data from the Kinects

2.4.7 Microsoft Kinect SDK

This is the software provided by Microsoft to interact with data captured by the Kinect. The SDK includes the driver to run the Kinect devices and act as the intermediate layer for users to reach the built-in data of the Kinect such as Skeletal and Joints data so that it can be processed with the Point Cloud Library and rendered by openFrameworks.

CHAPTER 3

EXPERIMENTAL SETUP AND TESTING RESULTS

This chapter discusses achieved results of the project. There are two types of results: one for the 3D Reconstruction problem and one for the 3D Data Streaming problem.

3.1 Experimental System Setup

Four Kinect devices are positioned so that every two Kinects are located 90 degree from each other and have a view of the same center location as shown in Figure 12. These Kinect 360s are responsible for capturing depth data. Another Kinect One is set up to capture skeletal data. After capturing, one single computer will handle the procedures for 3D Reconsctruction and 3D Streaming.

Figure 12. Experimental Setup.

3.2 3D Reconstruction Result

This section includes the results of the smaller problems from the 3D Reconstruction part, which consists of the following, data registration result, mesh generation result, and skeletal rigging result. The overview of the 3D reconstruction pipeline is described in Figure 13. This pipeline will take data from the 4 Kinect positioned like in Figure 12. The detail of the reconstruction pipeline consists of the following steps:

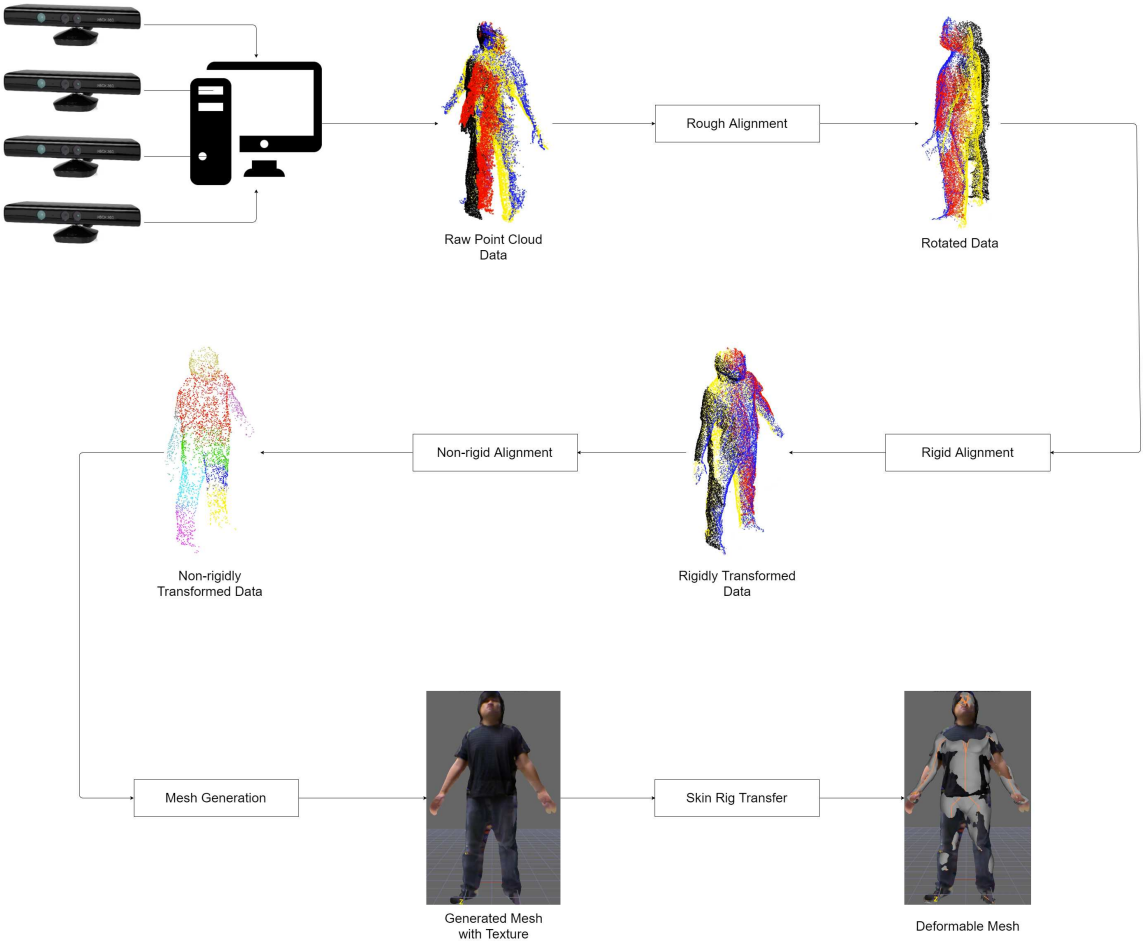

Figure 13. Reconstruction and Rigging Pipeline.

- Rough Alignment: raw data from four Kinects are rotated 90 degree from each

other.

- Rigid Aligment: Contour-based registration is applied on every two frames to minimize the difference between each set of point cloud.

- Non-rigid Alignment: The data are separated into 9 smaller body parts and contour-barsed registration is used again to minimized the difference between point cloud that belong to different body parts.

- Mesh Generation: Poisson surface reconstruction is applied on the calibrated data to create a 3D mesh with texture.

- Skin Rig Transfer: the high quality skeleton rig is taken from a SCAPE model and copied to the created mesh to create a deformable character mesh.

3.2.1 Data Registration Result

Figure 14 shows the captured data from each Kinect. According to the setup, each

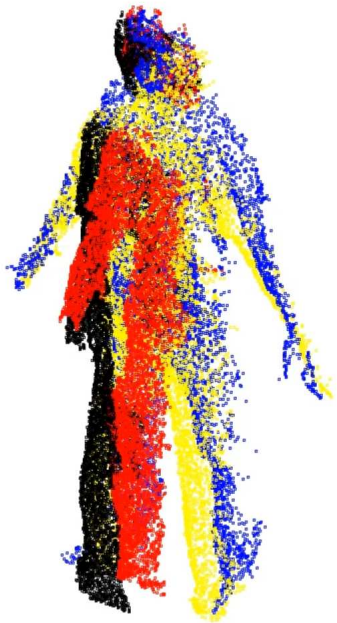

Figure 14. Unregistered Depth Data from 4 Kinects.

point cloud is rotated 90 degree from each other initially. Each scanned set of point cloud is color coded to enhance visualization.

The initial registration start with rotating depth data 0, 90, -90, and 180 degree respectively to the frame's setup location. The front frame will require no rotation. The right frame is rotated 90 degree clockwise. The back frame is rotated 180 degree. The left frame is rotated 90 degree counter-clockwise. The result of this step is illustrated in Figure 15. The data is roughly in their correct positions.

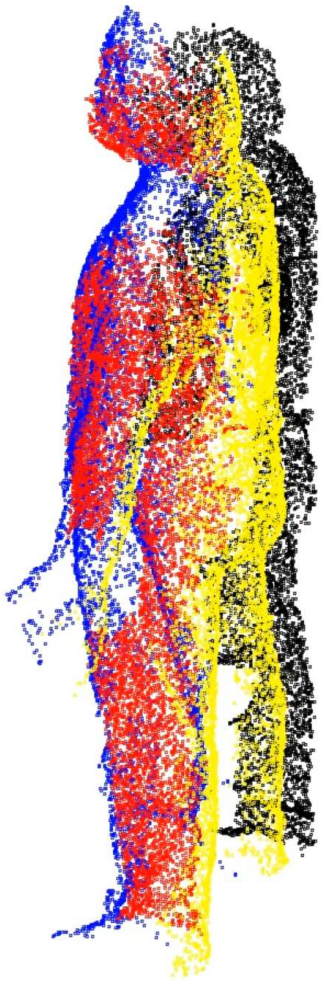

Figure 15. Initial Rotations.

The next step is minimizing the difference between frames using the contour-

based registration method. Figure 16 shows the result of this rigid regitration step. The point clouds are almost aligned, but the smaller body parts are still disjointed due to the sensor error.

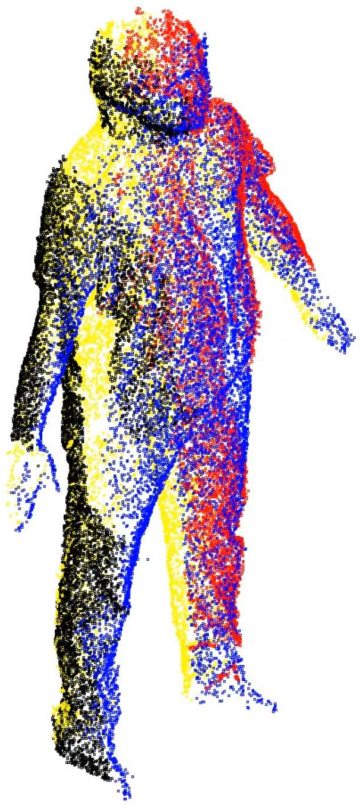

Figure 16. Rigid Registration.

To solve the smaller disjoints, the whole body is segmentated into 9 smaller body parts as seen in Figure 17. Then the contour-based registration is applied again, but on matching body parts on each frame. The final result is that all the point cloud data is now registered correctly.

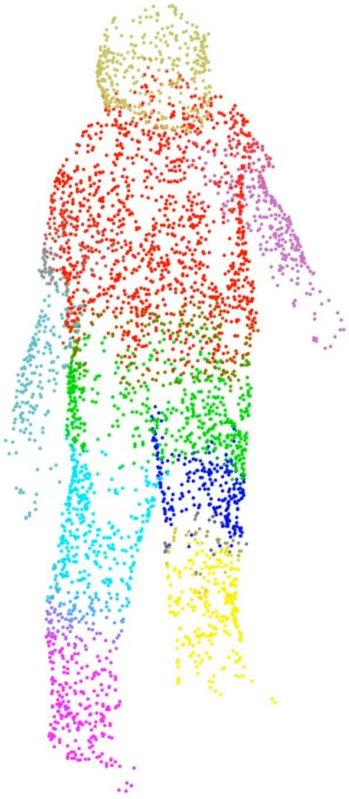

Figure 17. Non-Rigid Registration.

3.2.2 Mesh Generation Result

After applying the Poisson surface reconstruction method on the registered but unordered point cloud data, the final result can be seen in Figure 18.

3.2.3 Skeletal Rigging Result

The aformentioned automatic rigging method would fit the prepared SCAPE model with high-quality skeletal data into the generated 3D model and transferred the skinning rig to it. Figure 19 describe the fitting process. The output of this step is the full controllable 3D character.

Figure 18. Mesh Generation Result.

3.3 3D Streaming Result

This section includes the results from the solving partial problem of 3D Streaming, which are 3D model streaming, skeletal data streaming, and character animation. Figure 20 shows the streaming and character animation pipeline of the system. The following list consists of the main functions that the system do to perform 3D streaming:

- Model Transfer: The server will give the generated and rigged character from the reconstruction pipeline to the client.

- Skeletal Data Stream: Another Kinect One is connected to the system to capture the skeletal data. This data is packed into OSC messages and streamed over to

Figure 19. Automatic Rigging by fitting SCAPE model into the generated 3D mesh.

the client via OSC protocol.

- Transformation Apply: The transformation is computed based on the received joints data and applied to the character.

- GL Render: The system will render the rotated and translated model to the screen will OpenGL.

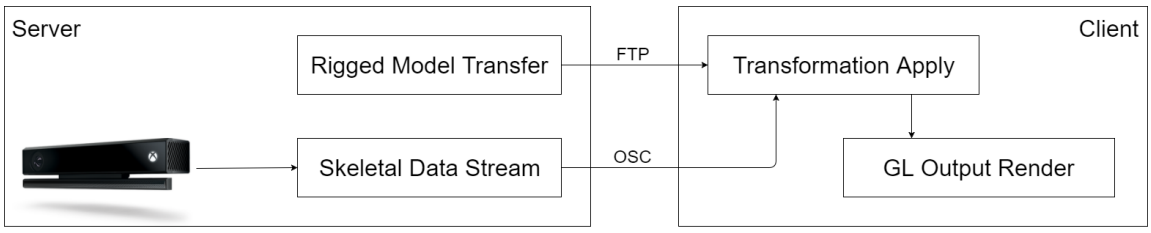

Figure 20. Streaming and Rendering Pipeline.

3.3.1 3D Model Streaming

After generation, the 3D model is given to the client by the server through a standard file transfer protocol. The user's own avatar and the other participant avatar can be rendered into one single space as shown in Figure 21

Figure 21. Mesh Transferred and Rendered.

3.3.2 Skeletal Data Streaming

By converting the skeletal data provided by the Kinects to OSC messages, it is possible to stream the skeletal data over the network using OSC. The program is open on a computer is fed with the Kinect Skeletal data and open a streamer. Another computer which acts as client will have the OSC Bridge ready to listen to the incoming

data from the streamer and display as seen in Figure 22

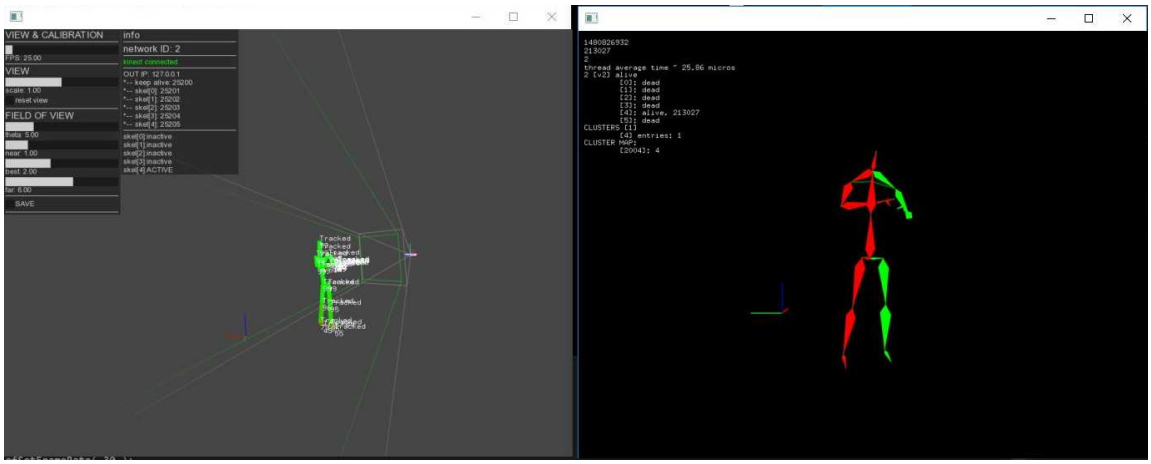

Figure 22. Skeletal Data Streaming.

Figure 23 shows the constructed and transferred OSC message that was received by the OSC Datamonitor program. The message content is the joints coordinates captured by the Kinect as float value. In actual test setup, the messages are received by the client.

Figure 23. OSC Message with Skeletal Data.

3.3.3 Character Animation

Figure 24 shows the rig information that was transferred from the SCAPE model in the rigging process. It is visible that the structure and the number of joints are

more complex than the Kinect's skeletal data in Figure 23, which only covers 25 joint locations.

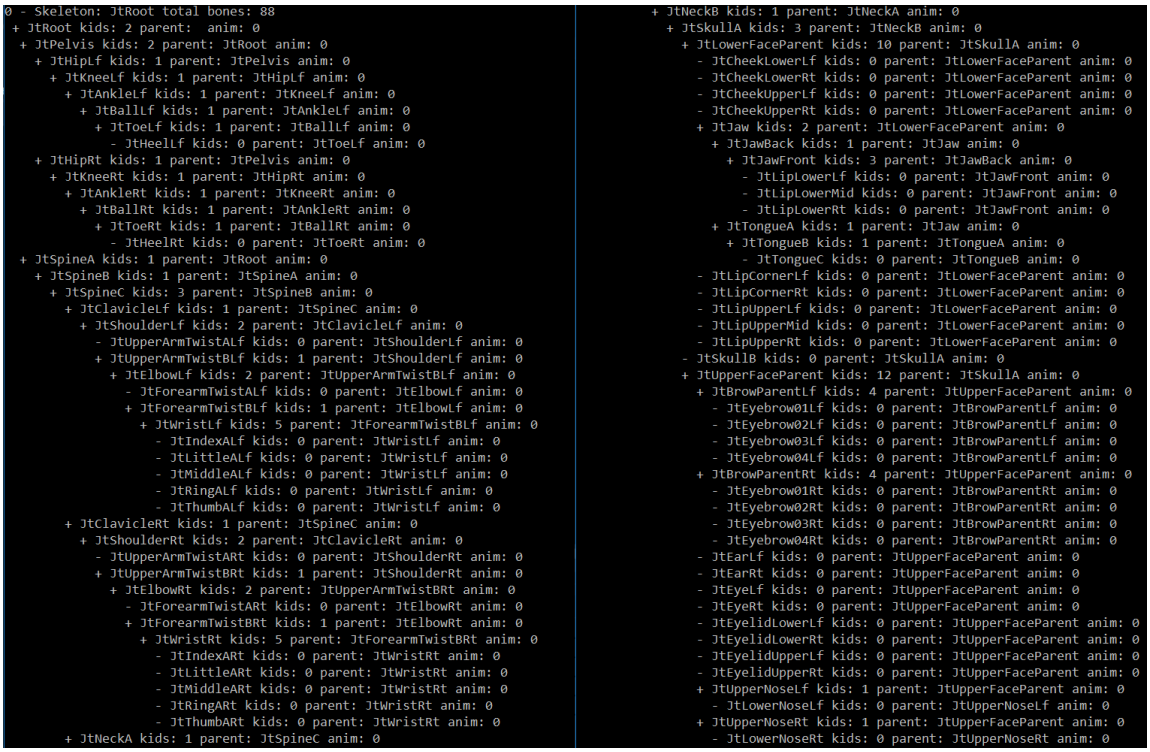

Figure 24. 3D Model Skeletal Rig Structure.

To animate the character with the Kinect data, it is necessary to match the joints from the 3D model with Kinect joint data. This process is called animation retargeting, which is usually done for every new 3D rig that is introduced into the system for the first time because other rigging method may return different skeletal structure. For this specific system, since all the skeleton rig are transferred from the same source dataset, the retargeting only needs to be done once. After that, the computations of the joint orientations are executed to animate the characters. Figure 25 shows the simulation of the virtual interaction between two users inside a virtual space.

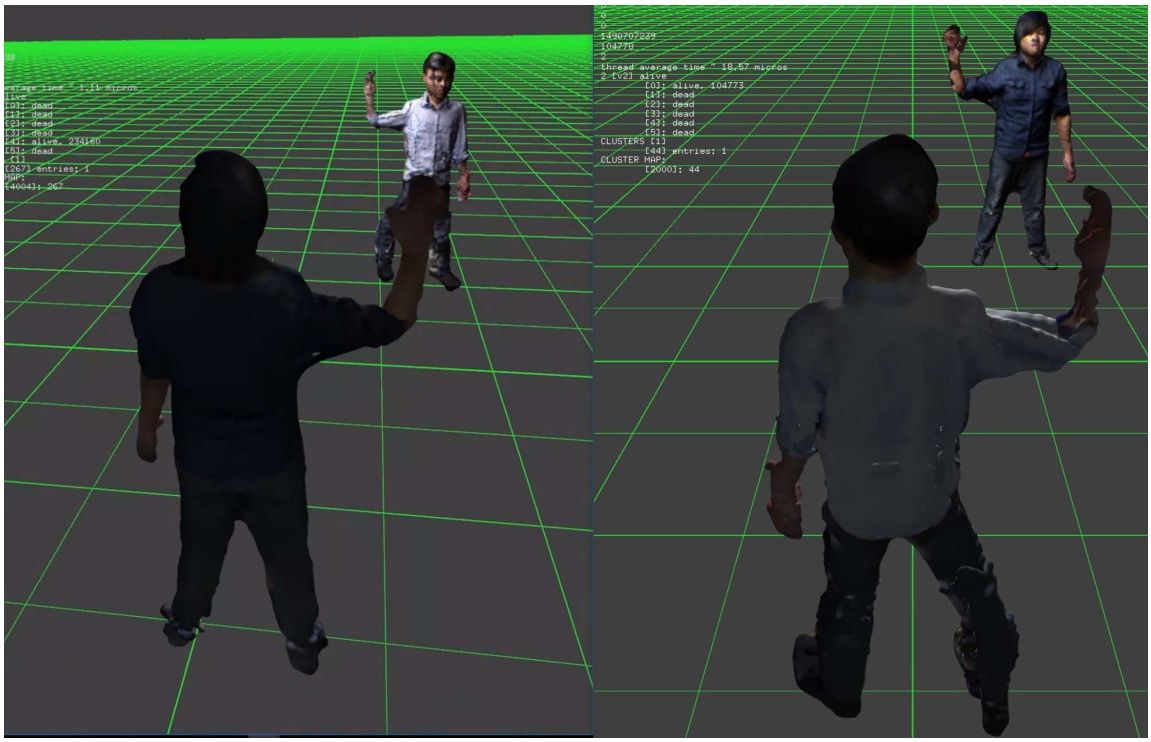

Figure 25. Virtual Interaction.

3.4 Result Evaluation

The created system is able to perform functions which were expected from the proposal. The system is geared towards the low-cost aspect. Specifically, hardware wise, any computer system with averagely sufficient performance can run the program. The system which were used for testing has the following specification, Intel Core i5 6600K, 16GB of RAM, AMD R9 390x GPU. This computer setup cost approximately $500, which is close to a middle-range PC. The depth devices consist of 4 Kinect 360 and 1 Kinect One, combining with their required PC adapters have the total cost of about $300. The final price of a whole system is less than $1000, which is still affordable compare to an enthusiast computer.

Software wise, after setting up, the program is able the handle the following func-

tionalities:

- Data Registration from connected depth devices
- Mesh Generation from registered data
- 3D Character Rigging from the created mesh
- Stream the created and rigged mesh to the client machine
- Stream the captured skeletal data to client
- Animate the character using the streaming skeletal data with kinematics
- Render the output data on the client side.

3.4.1 Data Registration from connected Kinects

The system is able to capture the depth data from 4 Kinects simultanenously and perform the two-step registration on the data. The first step of registration is the rigid transformation based on the contour-based registration. This step will minimize the difference between every two Kinects' frame. The second step of the registration is the non-rigid transformation. This step is performed by segmentating the whole body into smaller body parts in a heuristic fashion. By then, the contour-based registration is applied to each body part pair.

The output would be the point cloud data with set of 9 transformation matrices for each body part and the weight of each point for every matrix. The points which completely belong to the one body part will have the weight for the respective transformation matrix equal to one and the rest are zero. The points that lie in between body part will have weighted transformation. The final result of the registration process is a set of point cloud that is spatially correct.

3.4.2 Mesh Generation from registered data

This step ultilizes the Poisson mesh reconstruction to generate a tight mesh from the registrated data. The data from every Kinect also comes with color for each point, this is due to the RGB data of the Kinect has already been calibrated to the depth data. The Poisson texture blending algorithm is also applied to fill out the gaps and holes. The final out put is a fully 3D reconstructed model of a humanoid target. The whole process of capturing, registration, and mesh genaration takes approximately 5 minutes for the computer with listed specifications. It can be thought that the time required would be improved if the program is executed on a higher-spec machine.

3.4.3 3D Character Rigging from the created mesh

The automatic rigging process is executed based on a SCAPE model. The SCAPE model is morphable that allows change in both pose and body shape. The procedure of this automatic rigging method is also consists of two steps, fitting the deformable model into the 3D scan and transferring the skin rig to it. The first step compute the triangle correspondences between two model and deform the SCAPE model's vertex position to match the 3D scan. The second step copy the skinning rig, which include the joints data and skin weight to the 3D scan using the calculated correspondence from the deformation transfer step. The final result of this process is the high quality rigged and deformable 3D model that is the same as a referenced SCAPE model.

3.4.4 Stream the created and rigged mesh to the client machine

According to the proposed system flow, the generated 3D character needs to be given to the client by the server. As the created and rigged 3D model has size of a few megabytes, this step is done via a straight-forward file transfer protocol (FTP). With

the current standard of the network, the transfer process should not take more than a few minutes. The experimental setup, which is setup as a closed network, only required a few seconds for the file transfer. This step only need to be done once for each model. The client would store the character and ulitize it at later stage.

3.4.5 Stream the captured skeletal data to client

Another Kinect One is attached to the system to capture the skeletal data. The reason is the Kinect One is a product of better machine learning process that can return the skeletal data with more accuracy compare to the Kinect 360. The skeletal data from the Kinect consists of 25 joints' coordinates. For every resulted frame, these coordinates are bundled into an OSC message and sent to the client over the network. As each message is a combination of 25 float numbers as string, the time required to send a single message is insignificant. For the tested setup, from the server to client, approximately 60 OSC messages can be transferred per second. That would be enough to animate the character at 60 frames per second.

3.4.6 Animate the character

Using the generated 3D charater and the acquired skeletal data, the system can animate the character to move like the skeleton. This "puppeteering" step is done by computing the relative angle from the parent to the child joints, starting from the root of the skeleton, which is the pelvis. As the character skeletal rig is transferred from a high quality SCAPE model, the rig usually contains a number of joints that is much more than the 25 joints from the Kinect. The system is programmed to match the corresponding joints returned by the Kinect and joints from the character before calculating the rotation. The system can also apply locomotion transformation to

the charater.

3.4.7 Render the output data on the client side

The system is programmed using openFramworks as the base, which is capable of 3D rendering with openGL. As the generated 3D model is a Collada format, which is one of the standards for the industry, after calculating and applying the transformation to the model, the program can render the transformed model to the screeen. There is two rendered model in the program's virtual space, one is the current user of the system and the other is the received character that represents the other user. Two model rendered in a single virtual space at real time to simulate the virtual interactions.

CHAPTER 4

CONCLUSION AND FUTURE WORK

This is the final chapter of the project. This chapter provides potential future research and conclude this work.

4.1 Future Work

This section is based on the result evaluations. The section introduces the possibilities for the upgrades of the system in the future research.

4.1.1 Depth Data Registration

The registration for this project is currently done based on the assumption that the registration process will be executed for every 3D character creation. This is due to the physical locations of the sensors and the user position are subject to change. However, assume that the positions of the sensors are fixed, the calibration process can be sped up by reuse the last registration result. The calibration time can be significantly reduced from the current 2 to 3 minutes.

4.1.2 Skeletal Data Reading

The Kinect have already come with the built-in mechanism to return the stable joint data. However, the actual output of the joints is still jaggy. The result is that the animation of the character is also affected by the noisy data. This could be improved by introducing a filter into the system to smooth out the joints data. One possible downside is that more overhead and delay might also be added to the system.

4.1.3 User Recognition

The current pipeline is that the user create their avatars followed by the system animates that specific avatar. After one seesion, the character will be stored to the harddrive. Currently, the user has to choose their created avatar among the created models or recreate one from scratch. The system should employ some methods of recognizing if the user is already recorded within the system so that this process can be automated for future sessions.

4.1.4 Character Modular Approach

The system treats the user character as one single entity. This can be improved with the modular approach, which means the avatar of the user can be modified to certain extent, like changing clothes of the avatar. The users would be able to change their appearance as they wish. This can also reduce the need of the recreating the avatar for every session since the users would likely to use the same piece of clothing. The reconstruction process is only required for completely new elements which are introduced to the system.

4.1.5 Additional Data Stream

Along with the skeletal stream for the avatar controlling, the additional stream can be combined to enhance the user experience such as changing in surrounding environments or virtual objects with interaction. Even though this is a significant improvement in user experience, noticable increase of bandwidth requirement and overhead would pose a problem if it not addressed appropriately.

4.2 Conclusion

This work proposed an approach to create a cost-effective system that takes the current consumer grade depth camera such as Kinects to reconstruct 3D model and stream the data in real time. The system took inspiration from the novel idea of having holographic conversations which are shown in science-fiction movies, in which people can have conversation in person no matter the geographical barrier. The basis of the system is to create the avatar of the user, have the user control the avatar and stream the strange over the network to the client. This way, the streaming data does not require high bandwidth, but still able to show what the user is doing on the streaming side. Starting with the skeleton, more data can be streamed to enhance the viewing experience, such as facial data, or complex hand movements. Ultimately, the avatar will act the same as what the user is doing. Combining with virtual reality or augmented reality solution, an experience of having a virtual conversation from anywhere can be achieved.

REFERENCES

[1] Baran, I., and Popović, J. Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (July 2007).

[2] Boukerche, A., and Pazzi, R. W. N. Remote rendering and streaming of progressive panoramas for mobile devices. In *Proceedings of the 14th ACM International Conference on Multimedia* (New York, NY, USA, 2006), MM '06, ACM, pp. 691–694.

[3] Chao, Y., Wu, T., Wang, X., and Zheng, G. The computation of delaunay triangulation of lidar point cloud based on gpu. In *2015 23rd International Conference on Geoinformatics* (June 2015), pp. 1–4.

[4] Feng, A., Casas, D., and Shapiro, A. Avatar reshaping and automatic rigging using a deformable model. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games* (New York, NY, USA, 2015), MIG '15, ACM, pp. 57–64.

[5] Google. It's your turn. build the future., 2016.

[6] Intel. Overview of intel realsense sdk, 2016.

[7] Isenburg, M., and Lindstrom, P. Streaming meshes. In *VIS 05. IEEE Visualization, 2005.* (Oct 2005), pp. 231–238.

[8] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium*

on User Interface Software and Technology (New York, NY, USA, 2011), UIST '11, ACM, pp. 559–568.

[9] Jarrar, R. G. Image-based rendering protocols for remote interactive walkthroughs on mobile devices. Master's thesis, University of Ottawa, Canada, 2009.

[10] Kazhdan, M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70.

[11] Maamar, H. R., Boukerche, A., and Petriu, E. Streaming 3d meshes over thin mobile devices. *IEEE Wireless Communications 20* (June 2013), 136–142.

[12] Marton, Z. C., Rusu, R. B., and Beetz, M. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Kobe, Japan, May 12-17 2009).

[13] Ortiz, A., Oyarzun, D., Aizpurua, I., and Posada, J. Three-dimensional whole body of virtual character animation for its behavior in a virtual environment using h-anim and inverse kinematics. In *Proceedings Computer Graphics International, 2004.* (June 2004), pp. 307–310.

[14] Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., Kim, D., Davidson, P. L., Khamis, S., Dou, M., Tankovich, V., Loop, C., Cai, Q., Chou, P. A., Mennicken,

S., Valentin, J., Pradeep, V., Wang, S., Kang, S. B., Kohli, P., Lutchyn, Y., Keskin, C., and Izadi, S. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (New York, NY, USA, 2016), UIST '16, ACM, pp. 741–754.

[15] Perevalov, D., and Tatarnikov, I. S. *openFrameworks Essentials*. Packt Publishing, 2015.

[16] Rusinkiewicz, S., and Levoy, M. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (2001), pp. 145–152.

[17] Rusu, R. B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[18] Rusu, R. B., and Cousins, S. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, May 9-13 2011).

[19] Schmeder, A. Everything you ever wanted to know about open sound control. Tech. rep., 03/2008 2008.

[20] Seo, H., Joslin, C., Berner, U., Magnenat-Thalmann, N., Jovovic, M., Esmerado, J., Thalmann, D., and Palmer, I. Vpark - a windows nt software platform for a virtual networked amusement park. In *Proceedings Computer Graphics International 2000* (2000), pp. 309–315.

[21] Shapiro, A., Feng, A., Wang, R., Li, H., Bolas, M., Medioni, G., and Suma, E. Rapid avatar capture and simulation using commodity depth sensors. *Computer Animation and Virtual Worlds 25*, 3-4 (2014), 201211.

[22] Stanev, D., and Moustakas, K. Virtual human behavioural profile extraction using kinect based motion tracking. In *2014 International Conference on Cyberworlds* (Oct 2014), pp. 411–414.

[23] VR, O. A closer look at oculus avatars, oculus first contact, and asynchronous spacewarp, 2016.

[24] Zarean, A., and Kasaei, S. Human body 3d reconstruction in multiview soccer scenes by depth optimization. In *2016 24th Iranian Conference on Electrical Engineering (ICEE)* (May 2016), pp. 1591–1596.