i3DTI: INTERACTIVE 3D TELE-IMMERSION

by

Suraj Raghuraman

APPROVED BY SUPERVISORY COMMITTEE:

B. Prabhakaran, Chair

Xiaohu Guo

Ryan P. McMahan

Kang Zhang

Dedicated to my dad

i3DTI: INTERACTIVE 3D TELE-IMMERSION

by

SURAJ RAGHURAMAN, BE, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2017

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. B. Prabhakaran, for putting up with me. I would have given up a long time ago, if not for his continuous support and motivation. I would like to thank Dr. Guo and Dr. McMahan for the informative discussions, and Dr. Zhang for serving on my committee.

I would like to thank all of the people below for the following reasons:

- Gaurav for all of the discussions, data capture sessions, and particularly for his continued motivation and encouragement. I am also grateful for all of the food he fed me throughout the years.

- Kathy for being patient and persistent. I appreciate all of the hours she spent listening, questioning, and forcing me to finish, for whatever deadline we were up against. Thanks Kath, for typing large volumes, revising countless drafts (and all those last minute changes), and especially for doing those 2-3 day "last minute" runs before deadlines, without food or sleep, all so I could graduate. It is the "doer" in her that kept me going at the times I needed it the most.

- Raji for providing me with outside work, keeping me well fed, and enabling me to be financially independent.

- Karthik for the paper writing, diagrams, discussions, and of course, all of the parties.

- Arvind for the discussions, last minute motivational talks, and for editing many papers.

- Rajiv and Arjun for showing me why working full time is a bad idea for me.

- Kanchan for showing up everyday, and for being dependable and persistent.

- Kevin for always showing up for work and doing what needed to be done (no matter how mundane). He did a great job managing the lab and keeping it going.

- Brian and the CS Tech for helping out with the computer and network setups, especially with all of the issues associated with setting up the external gateway for the VA project.

- All the participants and users of the system, for working thorough all of the defects, evaluating the system, and giving helpful suggestions to improve the system.

- Shanthi for constantly reminding me how lively the environment was when I was in the lab.

- Rittika for all of the treats she left on the table, and for making me feel "happy to be alive" after riding in a car with her.

- Yuan for being as lazy as I am, while still showing up to the lab.

- Kira for doing the user studies without complaining, and the enthusiasm she had while doing it.

- Myunghoon for all of the kid stories, and showing me how to follow a schedule.

- Rahul for the ski trips, skydiving, and for taking my mind off of work.

- Sudhir for the wonderful scenes in unity, and the mountain biking trips.

- Cameron for switching to Unity, and highlighting the flaws in the naming conventions.

- Wang for helping me start the coding on the initial HUNA framework, and for following my coding style.

- Ganesh for highlighting the issue of storing large amounts of data.

- Sandeep, Abhinav, Ankit, Sapan, Dharmesh, Shantanu, and my family for persistently asking me "When will you finish your PhD?".

I would like to thank my family for motivating me enough to study abroad; if it weren't for them, I would still be working in India. I want to thank all of them for supporting me when I needed it the most. I would like to thank my friends for keeping me well fed, making me happy during the bad times, and for motivating me enough to get me to (finally) finish my dissertation and get my degree. If it weren't for all of you, I would have never even thought that finishing my PhD was possible.

February 2017

i3DTI: INTERACTIVE 3D TELE-IMMERSION

Suraj Raghuraman, PhD
The University of Texas at Dallas, 2017

Supervising Professor: B. Prabhakaran, Chair

3D Tele-Immersion (3DTI) approaches offer collaborative augmented virtuality, by immersing multiple geographically distributed users in a single virtual world. Multiple cameras are used to capture remote users and reconstruct their 3D models, to be rendered in the virtual world. Large volumes of noisy data, captured by the cameras, are cleaned, processed, transmitted, and rendered every frame. The complexities in processing and large transmission payloads result in high latency, low frame rate rendering, leading to very limited interactions between the users of the system. Interactive 3D Tele-Immersion (i3DTI) systems allow geographically distributed users to be immersed in highly engaging and interactive virtual worlds.

This dissertation presents a set of novel approaches, that improve the quality and performance of all of the stages of an i3DTI application. An image based meshing approach reduces the time taken for the 3D reconstruction of the captured data, to less than a millisecond. To ensure low latency even over the internet, a skeleton based prediction strategy is presented, that reduces the quantity of data transmitted per frame to just a few hundred bytes, while still maintaining good quality rendering. Naturalistic full body interactions, based on the skeleton that is estimated using multiple RGB-D cameras, keeps the users engaged while using the system. Current 3DTI systems use internal clocks to measure the latency experienced by the users, while ignoring the time for capture, rendering, screen refreshing, etc.

Two novel, millisecond accurate approaches, for measuring the latency felt by the user, are presented. One of the approaches measures the latency across the internet while the system is not actively in use, and the other measures the local latency while the system is in use. All of the approaches are implemented as a framework in a highly scalable, performance optimized, easy to use, and extendable architecture. Multiple applications, catering to domains from Tele-Medicine to education, were created using the framework. The patient trials of a remote patient diagnosis system, that was implemented between two geographically distributed locations using the i3DTI framework, was highly appreciated by the users; and the remote patient diagnosis was highly correlated to the in-person diagnosis.

TABLE OF CONTENTS

ACKNOWLEDGMENTS v
ABSTRACT viii
LIST OF FIGURES xvi
LIST OF TABLES xxiii
CHAPTER 1 INTRODUCTION 1
1.1 3D Tele-Immersion 1
1.2 Interactive 3D Tele-Immersion 2
1.2.1 i3DTI Minimum Performance Requirements 3
1.3 Contributions 4
1.4 Dissertation Overview 5
PART I LOOK AND FEEL 7
CHAPTER 2 CALIBRATION 9
2.1 Related Work 10
2.2 Ball Calibration 12
2.2.1 Point Selection 12
2.2.2 Scene Calibration 17
2.3 Evaluation 20
2.4 Discussion 23
CHAPTER 3 RENDERING 25
3.1 Related Work 26
3.2 Noise Removal 28
3.3 Segmentation 29
3.3.1 Person Extraction 29
3.3.2 Volumetric Segmentation 29
3.4 Image Meshing 30
3.5 Low Poly Reconstruction 32
3.6 Sparse Meshing 32
3.7 View Dependent Rendering 34

3.8 Rendering Overlapping Meshes . . . 36
CHAPTER 4 INTERACTION . . . 38
4.1 Skeleton Joint based Interactions . . . 38
4.2 Full Body Interaction . . . 40
4.3 Multiple Kinect Pose Detection . . . 42
4.3.1 Joint Accuracy based Consensus . . . 43
4.3.2 Kinect Skeleton Defects . . . 45
4.3.3 Probability of Accurate Joint . . . 47
4.3.4 Side Reassignment . . . 48
4.3.5 Consensus Skeleton . . . 50
4.3.6 Person Direction . . . 51
4.3.7 Skeleton Accuracy . . . 53
4.3.8 Implementation . . . 56
4.4 Body Sensor Enhanced Skeleton . . . 58
4.5 Haptic Enhanced Skeleton . . . 59
PART II COMMUNICATION . . . 62
CHAPTER 5 SKELETON PREDICTION . . . 64
5.1 Our Approach . . . 66
5.1.1 Skeleton Identification . . . 66
5.1.2 Segmentation . . . 68
5.1.3 Prediction . . . 69
5.2 Implementation . . . 70
5.3 Experiments . . . 71
5.3.1 Comparison With Existing Approaches . . . 73
CHAPTER 6 DISTORTION SCORE BASED POSE SELECTION . . . 75
6.1 Related Works . . . 78
6.2 3D Tele-Immersion Overview . . . 79
6.3 Visual Quality Challenges . . . 81
6.4 DISPOSE . . . 83

6.4.1 Formal Definition of the Pose Selection Problem . . . 84
6.4.2 Distortion Score . . . 85
6.4.3 Pose Selection . . . 88
6.4.4 Animation Sequences . . . 89
6.5 DISPOSE in 3DTI . . . 90
6.5.1 Sender Side - Identifying Candidate Pose . . . 91
6.5.2 Candidate Pose Repository . . . 92
6.6 Evaluation . . . 94
6.6.1 Experiments . . . 95
6.7 Discussion . . . 99
CHAPTER 7 SECURITY IMPACT OF PREDICTION . . . 103
7.1 Anti-Forensic Framework . . . 105
7.1.1 Real-time Segmentation . . . 106
7.1.2 Behavior Manipulation . . . 108
7.2 Evaluation . . . 110
7.2.1 Forensic Evaluation . . . 112
7.2.2 User Study . . . 114
7.3 Discussion . . . 115
PART III OBSERVED LATENCY . . . 117
CHAPTER 8 SCENE LATENCY . . . 119
8.0.1 Measuring Latency . . . 120
8.0.2 Outside Observer Technique . . . 121
8.1 VPLE . . . 122
8.1.1 The Approach . . . 123
8.2 Related Work . . . 124
8.2.1 3D Tele-Immersion Overview . . . 125
8.3 Pattern Generation . . . 129
8.3.1 Challenges . . . 129
8.3.2 Visual Pattern . . . 130

8.4 Pattern Recognition . . . 133
8.5 Latency Estimation . . . 136
8.6 VPLE in 3DTI . . . 137
8.6.1 Single Camera . . . 137
8.6.2 Multiple Cameras . . . 138
8.6.3 Multiple Sites . . . 138
8.7 Implementation . . . 139
8.8 Evaluation . . . 141
8.9 Experiments . . . 143
8.10 Discussion . . . 147
CHAPTER 9 ACTION LATENCY . . . 149
9.1 Related Works . . . 151
9.1.1 3D Presence . . . 152
9.2 Strobe Approach . . . 153
9.2.1 Strobe Signal Identification . . . 155
9.2.2 State Recognition . . . 156
9.2.3 Latency Estimation . . . 157
9.3 Implementation . . . 158
9.4 Evaluation . . . 159
9.5 Experiments . . . 161
9.5.1 Camera Latency . . . 161
9.5.2 Display Latency . . . 161
9.5.3 Single Camera . . . 162
9.5.4 Multiple Camera . . . 164
CHAPTER 10 VISUAL QUALITY VS RESPONSIVENESS . . . 165
10.1 Related Work . . . 167
10.2 i3DTI Framework . . . 168
10.2.1 Rendering . . . 168
10.2.2 Interaction . . . 170

10.3 The Game . . . 171
10.3.1 Game Requirements . . . 171
10.3.2 Penalty Shootout . . . 172
10.3.3 Mini Games . . . 173
10.3.4 Implementation . . . 174
10.4 User Study . . . 175
10.4.1 Mini Games Study . . . 176
10.4.2 Penalty Shootout Study . . . 180
PART IV APPLICATIONS . . . 182
CHAPTER 11 3D TACTILE TELE-IMMERSION FOR TELE-MEDICINE . . . 184
11.1 3DTTI Framework . . . 186
11.1.1 3D Presence . . . 186
11.1.2 Haptic Transparency . . . 187
11.2 Remote Physical Diagnosis System . . . 189
11.2.1 In Person Diagnosis . . . 190
11.2.2 Real World Challenges . . . 191
11.2.3 Motion Mapping . . . 192
11.2.4 Implementation . . . 194
11.3 Experiments . . . 196
11.3.1 System Performance . . . 196
11.3.2 Preliminary Study . . . 198
11.3.3 Patient Trials . . . 200
11.4 Discussion . . . 203
CHAPTER 12 OTHER APPLICATIONS . . . 205
12.1 Tennis . . . 205
12.2 Baseball . . . 209
12.3 Cardio-Pulmonary Resuscitation . . . 211
12.3.1 Challenges . . . 212
12.3.2 Our Solution . . . 212

12.4 HoloBubble 215
12.5 Exergames 216
12.5.1 ShotPut 217
12.5.2 Bowling 218
12.5.3 RehabQuiz 219
12.5.4 Balance 221
CHAPTER 13 CONCLUSION 223
APPENDIX A VISUAL QUALITY VS RESPONSIVENESS QUESTIONNAIRE . . 225
APPENDIX B REMOTE PHYSICAL DIAGNOSTIC SYSTEM QUESTIONNAIRE 228
REFERENCES 230
BIOGRAPHICAL SKETCH 239
CURRICULUM VITAE

LIST OF FIGURES

1.1 The real world user, captured using multiple Kinect V2 cameras (left), and rendered in a virtual world (right). 2

2.1 Reconstructed scene with a person captured with 7 Kinect V2 sensors, and calibrated using ball calibration. 10

2.2 Various stages of foreground extraction from left to right: original depth image, depth image after pre-processing, average background and the extracted foreground. 13

2.3 Circle Detection from left to right: edges in foreground image and circle detected using Circular Hough Transform. 14

2.4 Sphere fit from left to right: sampled points (green) from detected circle (red), points projected into 3D space, sphere fit for the points, viewed from front and top. 15

2.5 Calibration results between Kinect A (red) and Kinect B (blue), from left to right: without calibration and after calibration. 18

2.6 Direct registration of the points (left), and calibration by propagated registration (right), for two Kinect V2 sensors located at bottom right (red), and left middle (blue), with $5m$ of separation between them. In the scene, a ball is placed on the box next to a chair. 19

2.7 Head getting detected as ball when the Kinect V2 is placed high facing downwards (left), and wearing a hood over the head eliminates the false detection (right). 21

2.8 Calibration result starting from top left: our ball calibration, visual pattern approach (Kowalski et al., 2015), Zhang's method (Zhang, 2000), and plane intersection approach (Auvinet et al., 2012). 22

2.9 Point cloud rendering of a scene with multiple objects, captured using 7 Kinect V2 sensors, calibrated simultaneously using our ball calibration method from left to right: front and top view of scene. 23

2.10 Noise while capturing a plane (left), and a sphere (right), using a single Kinect V2 sensor located on the left side. 24

3.1 The processing pipeline for each of the cameras, in an i3DTI system. 26

3.2 An orthogonal rendering of a point cloud of a plane, captured using the Kinect V2 without any noise filtering (left), and the results of the use of median filtering (right). 28

3.3 Segmentation from left to right: captured scene, person extracted using region growing and volumetric segmented scene. 30

3.4 The square pattern used for triangulation from left to right: the selected points, the 2 triangles if all points are within range, and possible triangle options depending on the proximity of the points. 31

3.5 Meshing results: High resolution dense mesh created using the original depth image (left), and the lower resolution uniformly sparse mesh generated from a Gaussian scaled down depth image (right). 33

3.6 The steps involved in the sparse mesh generation approach. 33

3.7 The sparse meshing of the profile of a user, from left to right, $12,000$ vertices mesh, textured mesh, $1,000$ vertices mesh, and textured mesh. 34

3.8 Left: View dependent rendering based on the position of the virtual and real world camera. Right: A user model rendered in the scene using view dependent rendering. 35

3.9 The effect of rendering overlapping meshes without an alpha blended shader (left), and with a specialized alpha blended texture shader (right). 37

4.1 Joint based interaction from left to right: Picking up objects by closing the hand, and positioning the table tennis racket in the hand. 39

4.2 Skeleton based full body colliders for various users. 41

4.3 Overview of our skeleton estimation approach. 45

4.4 The camera on the right tags the right side of the skeleton (red) and the left (green); even after applying a camera extrinsic to change it to a left camera point of view, the joint is still incorrectly tagged, thereby requiring manual side reassignment. 49

4.5 Person direction estimation using the head and torso region: The front part of the head protrudes outwards more than the back, irrespective of looking forward (left) or sideways (right). 54

4.6 The containment score verifies that the skeleton is inside the person, by ensuring that the skeleton is behind the depth data, from all of the camera views. . . . 55

4.7 The capture area with 12 Kinect cameras, that was used in our experiments. . 57

4.8 Combining the Kinect skeleton with the bone information from the IMU sensor. 60

4.9 Hand joint estimation using the position of the haptic. 61

5.1 Shows the overall sequence of processing for the approach. 66

5.2 Skeletons and their corresponding mesh segmentation 67

5.3 Architecture of the 3D Tele-immersive system 70

5.4 Incorrect mesh segmentation and its corresponding skeleton for folded hands . 72

5.5 Prediction sequence based on the first frame for a one second period at 5 fps. Top row shows the actual data and bottom row the predicted result. 73

6.1 The architecture of DISPOSE based 3DPTI. 76

6.2 From left to right: The various artifacts *(highlighted)* that are generated by occlusion, rigging, segmentation, meshing and skinning, respectively. 81

6.3 The effect of using various scoring methods: *(left to right)* selected pose with holistic scoring, selected pose with regional scoring, animated result for holistic scoring(less than 40 triangles in the head region change), animated result for regional scoring. 87

6.4 Effect of selected pose on animation: T-pose *(left)* and up arrow pose. *(right)* Due to the stretching of the shirt, while in T-Pose, the results look considerably different. 91

6.5 Illustration of 12 fundamental poses captured in the dataset. 94

6.6 Instances of Varying Distortion Scores from multiple participants, in increasing order from left to right. 96

6.7 Comparison of processing times for the exhaustive search and threshold based search. 97

6.8 The mesh selected by the exhaustive search *(left)*, having $\sigma = 0.02$, and the threshold based selected mesh *(right)*, having $\sigma = 0.14$ 98

6.9 Results of the user study for preference of transmission strategies. 100

6.10 Capture with self occlusion: *(left to right)* captured mesh with occlusion, selected pose, and animated mesh without occlusion. 101

6.11 Incorrect skeleton detection, resulting in inaccurate animation: *(left to right)* detected skeleton, actual mesh, and animated mesh. 102

7.1 The behavior manipulated rendering of a person 104

7.2 The various aspects of anti-forensic framework to generate behavior manipulated streams. 106

7.3 The various stages in the segmentation process – (from left to right) the depth image is filtered to extract the person, then edges are estimated, skeleton is overlaid on the edge image, segments are then identified by growing from the seed points on the joint, reaching the final result. 107

7.4 The effect of number of seeds points on the segmentation, from left to right the skeleton is misaligned a little, using a single seed point at the middle of the joint results in bad segmentation and using the multiple seeds fix gives optimal segmentation. 108

7.5 Deformed meshes generated without using segmentation information are on the left, and with segmentation information on the right. 109

7.6 Altered color image generated without interpolation on top and with interpolation on bottom 111

7.7 The segmentation done using Voronoi on the left and edge based method on the right, notice the head region 112

7.8 Noise analysis for original depth images, Type I and Type II forged depth images with single image and multiple images as background in all frames. 113

8.1 A sample layout with a high speed camera (HSC) recording the displays for the VPLE based latency estimation of a system with four Kinect V2s, processed on camera machines (CM1 to CM4). 120

8.2 A sample layout, with a high speed camera (HSC) recording the displays for the VPLE based latency estimation application (App), with a Kinect V2. . . . 123

8.3 The 3DTI processing pipeline of a single site, with one RGB-D camera. 126

8.4 Issues with HFR, high exposure, and capture from monitors. From Left to Right: a grid captured with geometric distortion, the capture of a fast moving dot causing overexposure, the HFR capture of a color gradient grid shown on the monitor, the HFR capture of a black monitor, and flickering due to partial frame rendering. 129

8.5 From Left to Right (not to scale): The structure of the pattern, and the sequence of 2 consecutive patterns, generated using our approach at 144 Hz. 131

8.6 Patterns captured using the HSC from Left to Right: a generated pattern P_G, the corresponding pattern rendered by the application P_A with RoI marked in red, and the regions identified in P_A, with timer marked in yellow, and ET marked in blue. 133

8.7 A sample setup for latency estimation, in a multiple site 3DTI application using VPLE, by capturing the displays using high speed cameras (HSC). 139

8.8 Left to Right: Variation in σ growth based on display refresh rates, mismatch between σ of two synchronized pattern generators, and VPLE estimated $\mathcal{L}$ for simulated delays. 142

8.9 Implicit latencies for (from Left to Right): a single camera 3DTI, a multi-camera 3DTI and a multi-site 3DTI. 145

8.10 Observed latencies for (from Left to Right): a single camera 3DTI, a multi-camera 3DTI and a multi-site 3DTI. 146

9.1 The Strobe approach setup to measure a multiple camera 3D presence system, using a green strobe and a high speed camera (HSC). 150

9.2 The effects of μ on the estimated strobe signal. 155
9.3 The effect of an incorrect σ on the rectangular state wave. 156
9.4 The local latency (l) that is measured between the state waves, W_R and W_D. . 157
9.5 Various states of the strobe from top to bottom: Strobe turned On, turned Off, and the overexposed capture of On to Off. 158
9.6 3D reconstruction latency plots from left to right: Implicit latency measured for an entire scene, the implicit and observed latency for a person waving and moving backwards, away from the camera. 163
10.1 Third person view of the striker kicking the ball towards the goal in the penalty shootout game. 166
10.2 Reconstruction pipeline for a 3D immersive application using MS Kinect V2 for capture, and Unity3D for rendering. 169
10.3 Side view of the player mesh using view based rendering (left), and front and back mesh rendering (right). 170
10.4 Skeleton based colliders are shown in green for players of different size. 171
10.5 Play area setup, with multiple MS Kinect V2 cameras (marked in red) for capture, TV, and projector for rendering. 172
10.6 From left to right: Goalkeeper, with the circle showing the expected ball position, and the striker with practice targets. 174
10.7 Box plot of user ratings for OVQ study 178
10.8 Box plot of user ratings for OIQ study 179
10.9 Quantitative statistics of player performance for 1) OVQ and 2) OIQ. 179
10.10 Box plot of user ratings for the two player penalty shootout game. 180
11.1 The rendering from the patient side, captured during an RPDS session. 185
11.2 The steps involved in the 3D reconstruction of a person, captured using a MS Kinect V2 camera. 187
11.3 The haptic force feeback rendering for a two site transparency situation. 188
11.4 Arm motions for diagnosis shown top to bottom, left to right: elbow flexion, elbow extension, arm elevation, arm depression, shoulder internal rotation, shoulder external rotation, shoulder abduction, shoulder adduction, shoulder protraction, and shoulder retraction. 191
11.5 Arm motions adapted to the haptic device for diagnosis, shown from top to bottom, left to right: elbow flexion, elbow extension, arm elevation, arm depression, shoulder internal rotation, shoulder external rotation, shoulder abduction, shoulder adduction, shoulder protraction, and shoulder retraction. 193

11.6 The two site setup of RPDS, with the doctor at one site, and the patient at another. . . . 195

11.7 The system performance for the mesh (left) and the haptic (right) of the RPDS during a patient diagnosis session between the hospital and university. . . . 197

11.8 User study results for the preliminary study with healthy users (left), and 15 patients with shoulder discomfort (right). . . . 199

11.9 The close-up view of the patient, as rendered on the doctor's side, and captured during an RPDS session. . . . 200

11.10 User study results for the system specific questions, answered by the 15 patients. 201

11.11 The correlation between the in-person and RPDS diagnosis, of the patients in the study, from left to right: range of motion (ROM) and max isometric strength (MIS), green for match, red no match, and blue for match without considering pain; combined ROM and MIS (green means MIS and ROM match, red means none match, blue all match without pain, magenta ROM only match, yellow MIS only match. . . . 202

12.1 An overview of the tennis game setup, using two sites with a player captured using two Kinect V1 cameras on each site, a BSN sensor on each player's hand, and a projector displaying the game. . . . 206

12.2 The BSN sensor on the player's hand, and the corresponding orientation of the tennis racket in the game. . . . 207

12.3 The default third person view of a player playing tennis, showing both the players rendered as a point cloud. . . . 208

12.4 The pitcher finishing his pitch, with his arm in front of his leg, and the ball in the air. . . . 209

12.5 The batter, holding the bat, ready to hit the ball. . . . 210

12.6 The immersive CPR trainer. (a) shows the real-world scene, (b) shows the visual occlusion problem, (c) shows the virtual world scene. . . . 211

12.7 The working procedure of the immersive CPR trainer. . . . 213

12.8 HoloBubble virtual scene from left to right: the design layout, and a virtual rendering that shows the colored bubble generators and mirror setup. . . . 215

12.9 From left to right: The front view of the player playing the game, and the rendered view of the player captured from behind, with the mirrors showing the front and side of the player. . . . 216

12.10 The ShotPut game, associated with elbow flexion, shows a person throwing the 'shot' as far as possible, and the distance its thrown from the foul line is how the score is calculated. The score appears in the top right corner. . . . 218

12.11 The player practices the elbow extension exercise by playing the Bowling game, and earns more points for each pin they knock down. 219

12.12 A person using the elbow rotation exercise while playing the RehabQuiz, a trivia game; this shows how the player picked up a trivia card by closing his hand near the deck of cards, causing a trivia question to appear on the screen. 220

12.13 The Balance game, with the player practicing the hip abduction exercise by standing on one leg, on a plank floating on water. 222

LIST OF TABLES

6.1 Comparison of various 3DTI transmission strategies in literature. 99

CHAPTER 1

INTRODUCTION

All of the communication on the internet today is based on either text, audio, or video. Audiovisual communication restricts the users to a fixed visual point of view, while interacting with each other. This fixed point of view limits the level of user engagement, and the possible modes of communication between the users. Collaborative Augmented Virtuality (CAV) systems immerse multiple geographically distributed users into a single environment, that combines both virtual and real world elements. These CAV systems enable users to communicate among themselves, using newer modalities that go beyond the traditional multimedia modalities of text, audio, and video. With the continuous improvement of technologies like: head mount displays that provide a new way of viewing the virtual world, haptic devices that provide a sense of touch and force feedback, body sensors that can measure various aspects and also provide tactile feedback, digital scent technologies that can detect and recreate odors, and gustatory technologies that provide a virtual sense of taste, etc., CAV systems facilitate much greater degrees of interaction and create a far better experience for the users.

1.1 3D Tele-Immersion

3D Tele-Immersion (3DTI) systems try to actualize CAV, by capturing and transporting users from different locations into a single virtual world, where they are represented by their own "live" avatars, as shown in Figure 1.1. To show these "live" 3D models, 3DTI systems surround and capture the user, with multiple calibrated cameras. All of the cameras capture the user at a minimum of $30fps$, and in every frame, a 3D model of the user is reconstructed, using the captured data. The model is then transmitted over the internet, to all of the other users that are co-present in the 3DTI session. The 3D model of every user is rendered in the virtual world, that is being displayed to each user, creating an illusion of co-presence. On

Figure 1.1. The real world user, captured using multiple Kinect V2 cameras (left), and rendered in a virtual world (right).

average, a RGB-D camera, such as a Kinect V2, generates about 6MB of data per frame, for a good quality reconstruction; a typical 3DTI system uses at least 4 cameras, leading to about $24MB$ of data per frame. Transmitting $24MB$ of data per frame, at 30 frames per second, is challenging, even over a local gigabit Ethernet. The level of noise and the complexities of reconstruction, using traditional approaches, lead to a few 100ms of latency per frame, for just the processing alone. The combination of processing and transmission delays cause considerable latency for the users, triggering user dissatisfaction. The high latency, and heavy processing costs of 3DTI also restrict the rendering quality of, and user interactions with, the virtual world; these complications limit most 3DTI applications even further, by making the use or enjoyment of the system unrealistic for a majority of users. Thus, traditional 3DTI systems fail to accomplish the principles of CAV, while achieving an acceptable level of user satisfaction.

1.2 Interactive 3D Tele-Immersion

To better accomplish the principles of CAV while attaining a high quality user experience, the novel concept of interactive 3D Tele-Immersion (i3DTI) is introduced in this dissertation. An i3DTI system brings users, from different parts of the world, together into a highly interactive

virtual world. The endless creative renderings possible in the virtual world, combined with the superior level of interactions, deliver the user into a new world, where they star in a realistic live game, while the real world fades away around them. The users of the i3DTI systems interact with the virtual world naturally, using their entire body, just as they do in the real world.

1.2.1 i3DTI Minimum Performance Requirements

In order to provide the best possible user experience, an i3DTI application needs to at least meet, if not exceed, certain minimum performance requirements:

1. For a better viewing experience, the motion picture group recommends a minimum frame rate of 24 fps, for all videos containing speech and audio, and 16 fps for videos without audio. While i3DTI applications can exist, without necessitating any verbal interaction with the user, it is counter-intuitive to create such an application given the collaborative nature of i3DTI systems; i3DTI systems need to provide higher frame rates, of at least 24 fps for the use of "live" 3D avatars, while ensuring an above average viewing experience for the users.

2. According to the online gaming community, a ping of over $120ms$ leads to noticeable lag and a poor gaming experience. The i3DTI system need to have a latency of less than $120ms$, to keep the user engaged in highly interactive virtual worlds. exceptional

3. In order to maintain the high level of performance, the i3DTI framework should have a massively distributed and parallel architecture. The i3DTI framework should be easily expandable, allowing quick and easy additions of new algorithms, devices, modalities, etc., without the need for extensive code additions or rewrites. The i3DTI framework should be easy to use, and allow developers, with limited coding skills and experience, to be able to create an i3DTI application.

1.3 Contributions

An i3DTI system incorporates technologies from the fields of computer vision, machine learning, networking, computer graphics, parallel computing, real-time systems, computer architecture, software architecture, big data, queuing theory, etc. To improve the performance of the 3DTI system on a massive scale, while reducing the resource usage, requires completely new, innovative, and ingenious approaches at problem solving. The entire i3DTI system was completely broken down, in an attempt to identify potential pressure points and performance bottlenecks. Instead of relying on existing established techniques for performing standard tasks, methods were considered, or disregarded, primarily based on performance. This process of elimination resulted in the identification of some new problems, and emphasized the necessity for faster solutions to existing problems. While the focus was on accomplishing the goal of creating an i3DTI system, many of the novel approaches presented in this dissertation have theoretical foundations, and have wide ranging implications in their corresponding fields. The main scientific contributions of the dissertation are:

- A fast, accurate and robust calibration technique, that can calibrate multiple cameras simultaneously to a single common space. Minor modifications to the technique enable the cameras to be calibrated, with any visible sensor.

- A distributed parts based image domain 3D reconstruction and rendering approach that can render a 3D model of an entire scene, captured using multiple cameras within a millisecond. The representation of the 3D reconstruction problem in the 2D image domain, where even complex time consuming operations, like mesh decimation, are solved in less than a millisecond.

- A real time skeleton pose detection approach based on the joint estimations from multiple sensors. The probabilistic joint model that allows the use of any sensor that is providing joint estimation, to boost the Kinect skeleton accuracy.

- An accurately scaled collider based whole body collision model based only on the skeleton.

- A low latency over the internet prediction based communication scheme for transmitting live 3D models. The approach relies on previously received mesh, and on few bytes of indicator data every frame, to render reasonably accurate live 3D models.

- Better quality low latency over the internet skeleton based multiple mesh prediction scheme for transmitting live 3D models. Distortion Score is used to measure the artifacts associated with animating a mesh.

- An approach to generate visually realistic forged color, depth, and 3D mesh streams, containing manipulations of people's actions.

- A fully automated accurate visual latency estimation approach for geographically distributed captures to render systems.

- A fully automated accurate (within 1ms) estimation of capture to display the latency for a visual system, using a strobe light.

1.4 Dissertation Overview

The dissertation is logically organized into four parts with each of the parts containing multiple chapters. The first part, Look and Feel, details all of the approaches that are involved from the calibration of the cameras to the rendering of the live 3D user model at high frame rates, with very low latency and low resource utilization. The techniques that enables fast full body interactions with virtual world objects are also explained here.

The second part deals with data communication over the internet, and describes novel methods that can transmit and render a "live" 3D user model within milliseconds, even at network speeds of less than $1Mbps$. The possibility of creating malicious data streams using

a communication approach is examined in detail, and a forgery detection method is also presented.

The third part, Visual Latency, presents innovative methods for measuring the true visual latency that is observed by the user, while using the system both locally, and over the internet. We present a study that highlights the importance of visual quality and responsiveness to the users of an i3DTI application.

The final part, Applications, showcases a real world i3DTI tele-health application, and traces the evolution of the i3DTI framework, using a few selective applications that were developed using the framework, throughout the years.

PART I

LOOK AND FEEL

The most processing intensive part of the entire i3DTI system is the 3D reconstruction and rendering of the user in the virtual world, while allowing whole body user interaction every frame. The approaches described here focus on the high quality visualization, and realistic interactions for the user in the virtual world.

Chapter 2: The cameras used to capture the user need to be calibrated with each other, to ensure the accurate 3D reconstruction of the user. The ball calibration approach uses a spherical object to simultaneously cross calibrate all of the cameras in the scene. All of the cameras can be calibrated using the ball approach in a couple of minutes, compared to the hours taken by the other approaches.

Chapter 3: The data captured by the cameras contains the entirety of the scene, and is prone to noise. Filters and segmentation approaches are used to reduce the noise, and isolate the user from the captured images. The 3D surface reconstruction approach described in the chapter, simultaneously reconstructs the captured data, while reducing the number of triangles used for the mesh representation. This optimized representation reduces the size of the mesh, allowing for faster network transmission and higher frame rate rendering of the virtual world.

Chapter 4: i3DTI applications allow for natural full body interactions in order to maximize the user's level of engagement and overall satisfaction; by allowing the user to use only their body instead of external devices, like keyboards, mouse devices, etc., it simplifies user interaction, and makes the virtual world experience more realistic. Rather than relying on the complete 3D reconstruction of the person for these interactions, a skeleton based approach is proposed, to improve both the quality and responsiveness of the virtual world interactions for the user. In this chapter, we present multiple approaches that can be used to accurately estimate the skeleton of the person, using different modalities. The skeleton pose detection approaches all use Kinect cameras, and are augmented with more cameras, body sensors, or haptic devices to improve skeleton accuracy significantly, even in situations with heavy occlusion.

CHAPTER 2

CALIBRATION

3D capture and immersion technologies (Izadi et al., 2011; Raghuraman et al., 2012; Kurillo and Bajcsy, 2013) use multiple cameras to capture a scene, or object, for reconstruction. 3D reconstruction of real world objects requires capturing the object using multiple cameras. The availability of commodity RGB-D cameras, such as Kinect V2, that can capture range images, has become a lot more affordable, resulting in a lot of research in the field. Accurate reconstruction requires both intrinsic calibration of the camera, and the extrinsic calibration between cameras. While the intrinsic calibration remains unchanged for a given camera, the extrinsic calibration changes whenever the camera is moved.

There are many methods to calibrate color cameras, both intrinsically and extrinsically. These methods have been extended to work with RGB-D cameras, by calibrating the color and the depth sensors with each other. Since they use a plane of some sort to calculate the reference points, these methods can typically calibrate two RGB-D sensors at a time. Calibrating a scene with multiple cameras, requires repeating the same calibration procedures between all pairs of adjacent cameras. This repetitious process is often time consuming, and leads to error propagation between cameras when the entire scene is calibrated. Procedures like bundle adjustment, and using a single reference camera to calibrate all other cameras to, are used to overcome error propagation. In spite of these efforts, achieving accurate calibration is still difficult and time consuming.

The Ball Calibration (BC) method introduced in this paper, can calibrate a scene with multiple cameras together, accurately in less than a minute. BC uses only the depth image to calibrate, eliminating the error propagation from the color and depth sensor calibration. A ball, or spherical object, is used for calibration instead of the plane, to allow all the cameras in the scene to simultaneously capture the ball; this not only reduces the total time taken to calibrate all the cameras, but also reduces the propagation of calibration errors between

Figure 2.1. Reconstructed scene with a person captured with 7 Kinect V2 sensors, and calibrated using ball calibration.

cameras. The final calibration is estimated pair-wise between cameras, using (Umeyama, 1991). In situations where the cameras cannot track the ball at the same time, due to range or orientation reasons, a registration propagation strategy is proposed to minimize the calibration errors between the two cameras. BC was able to calibrate a scene with 7 Kinect V2 sensors in under 3 minutes, without any manual intervention, except for moving the ball. A scene reconstructed with the calibration of the sensors is shown in Figure 2.1. The main contributions of our work are: A sphere based point acquisition strategy for accurate registration. A method for using the acquired points to calibrate all cameras in the scene at the same time. A propagation strategy to allow calibration of distant cameras with lesser error propagation, due to miscalibration of individual camera pairs.

2.1 Related Work

There has been a lot of research in the field of camera calibration over the years. In this section, we primarily focus on work used to calibrate RGB-D cameras, especially the Kinect.

A checkerboard pattern based calibration method was introduced by (Zhang, 2000). The checkerboard pattern is printed on a plane and shown to the cameras. The method uses the points, at the corners of the white and black squares, to identify point correspondence between the cameras. For calibrating the Kinect, either the infrared image from the Kinect is used, or the color cameras are calibrated with each other, and the depth cameras are calibrated with the color cameras using the infrared image. Due to the various cross calibrations involved, the process is time consuming and susceptible to inaccurate calibration, requiring re-calibration at various steps. An extra overhead Kinect was used by (Maimone et al., 2012) to minimize the propagation error, by calibrating all the other Kinect sensors with the overhead Kinect.

An intersection of plane based calibration method was proposed by (Auvinet et al., 2012). A plane whiteboard is shown to the camera, and rotated in an orthogonal manner on all three axes. The plane is tracked using a color detector. The points on the plane are fit to a plane model, and the intersection of these planes is used as the reference point between cameras. This is a very time consuming process, requiring accurate color to depth calibration and plane rotation. The result depends on the angles between the cameras and the real world board plane used. Methods based on the corners of the board (Beck and Froehlich, 2015; Herrera et al., 2012; Li et al., 2013) have also been proposed, but are influenced by edge noise, leading to poor calibration.

A 3D reconstruction approach based on block patterns was proposed by (Kowalski et al., 2015). The block pattern is printed on a plane and shown to the cameras. The approach can calibrate multiple Kinect V2 cameras, by calibrating their individual color cameras. Since the method may not be accurate due to the cross calibration between color and depth, Iterative Closest Points (ICP) is used to calibrate the Kinect V2 cameras for better results. While this method takes less time to calibrate, its calibration is not very accurate.

A color to depth calibration for the Kinect V1 sensor was proposed by (Staranowicz et al., 2013). A spherical object is shown to the Kinect, and detected on both the color and depth

images. An ellipse is used to track the object in the color image, and a RANSAC detector is used to track the object in the depth image. The center of the circle (depth) and ellipse (color) are used for calibration. This method was shown to be very effective in calibrating the color and depth cameras of the Kinect. However, this method cannot directly be applied to multiple Kinect calibration, due to its planar point selection scheme.

2.2 Ball Calibration

Calibrating multiple depth cameras requires the identification of some point correspondence between each of them. The Ball Calibration (BC) uses a ball to attain the point correspondence required to calibrate the scene. Using the ball ensures that all cameras can visually see the object, as long as the ball is placed inside the common scene that is captured by all the cameras. The center of the ball is used as the correspondence point between the cameras, to calibrate them. The actual calibration is performed for all pairs of cameras, on a pair by pair basis, using Horn's method (Horn et al., 1988). In situations where the overlap of the capture area between 2 cameras is small, instead of relying on or waiting for a fixed number of points to be captured, the calibration is propagated between the 2 cameras, using the calibration results of the cameras in between them. At the end of the procedure, all cameras are calibrated with each other, either directly or by a calibration propagation, to yield accurate spatial reconstruction.

2.2.1 Point Selection

BC uses the center of the ball to identify point correspondences between cameras. In order to automatically identify and select these points, both of these things need to happen: the ball needs to be detected and its' center accurately estimated. Ensuring correspondence requires that the cameras capture the scene at the same time, and have the ball clearly visible. Since

Figure 2.2. Various stages of foreground extraction from left to right: original depth image, depth image after pre-processing, average background and the extracted foreground.

the Kinect V2 sensor does not allow triggered image capture, it is ensured that the images are at most one frame apart (approximately 30ms).

The depth image captured by the Kinect V2 contains a lot of salt and pepper noise, noises around object edges, and holes. The Kinect V2 sensor has a range of about $5m$ and an elliptical conic (Yang et al., 2015), resulting in extremely noisy sides; these can be eliminated using a bounding box. The 3D bounding box eliminates the noisy sides of the image and points beyond $5m$. To reduce the influence of these noisy patterns and other surrounding objects, on ball detection, we perform background subtraction. A simple aggregated background is estimated, using the initial depth images of each of the camera views. For every depth image (I), as shown in Figure 2.2, using the background image (B), a foreground mask (F) is estimated as $|B - I| > \tau$, where τ is the background threshold set at $50mm$. The foreground mask is then applied on the depth image to extract the foreground.

Ball Detection

Once the foreground has been extracted from the depth image, the total area of interest for ball detection is restricted to this region of the image. The edges of the foreground image are estimated using the canny edge detector. A low edge threshold is used to ensure that even the minute inner and outer edges are detected, as seen in Figure 2.3. The conics of

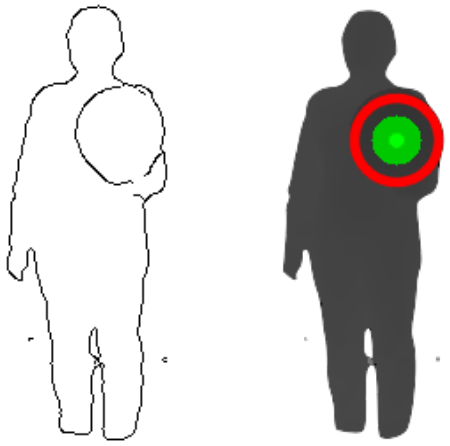

Figure 2.3. Circle Detection from left to right: edges in foreground image and circle detected using Circular Hough Transform.

the Kinect V2 sensor are largely uniform across the X- and Y- axes. Hence, circular objects inside the scene will remain circular in the image. For sensors having different X- and Y- axes, an image re-size would ensure that circular objects in the real world would remain circular in the image. A Circular Hough Transform (CHT) is applied on the edge image, to identify circular objects in the scene. CHT requires a range of possible radii for the circle, to reduce the amount of computation and accumulator storage. The size of the ball differs, depending on the position of the ball inside the scene. So when the approach is initialized, we assume that the ball is located in the range of 1 to 3 meters, resulting in a possible radii of 15 to 30 pixels. The circle detection threshold is set high, allowing even the low probability circles to be recognized. Since we know that there is only one ball inside the image, the circle with the highest probability is selected. For the subsequent frames, the range of the circle detection radii in CHT is kept at ± 5 pixels of the previously detected radii. Higher preference is given to the circle detected near the previous circle, from the previous depth image. Since the depth images are captured at $30 fps$, the ball cannot move a significant distance between consecutive depth images. Enforcing this restriction, in conjunction with sphere fitting, allows for accurate tracking of the ball.

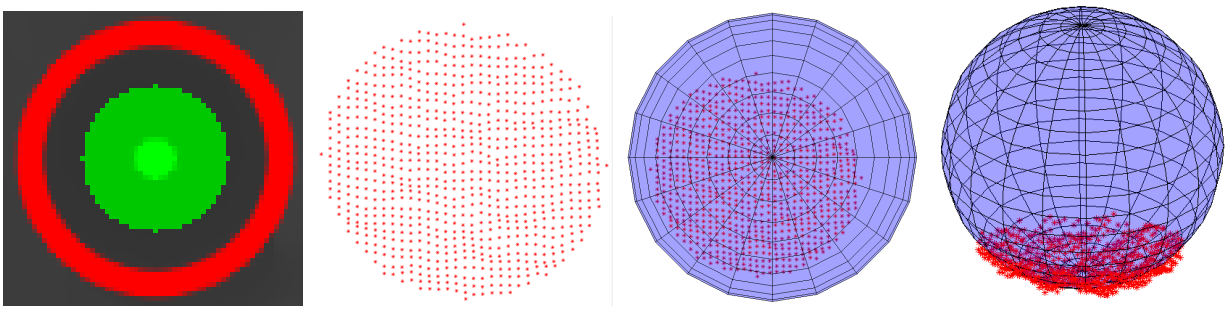

Figure 2.4. Sphere fit from left to right: sampled points (green) from detected circle (red), points projected into 3D space, sphere fit for the points, viewed from front and top.

Sphere Fitting

Once the circle has been detected in the depth image, we first sample the circle area to extract points, which would be part of the ball represented as a sphere in 3D. Since we want to avoid the hand, or other areas which are not part of the ball, a smaller sampling circle, typically 60% of the original radius, is used. By using just a fraction of the detected circle, we ensure that neither the edge noise, nor external entities (fingers, hands, etc), contribute to the sample point set. As seen in Figure 2.4, after the points are identified in the image, they are projected into the 3D world coordinate system, using the intrinsic calibration of the camera.

These points in 3D represent the side of the surface of the sphere. A least square fit approach is used to identify the center and radius of the sphere. A sphere is represented with points (x, y, z) on the surface, and can be written as:

$$x^2 + y^2 + z^2 + ax + by + cz + d = 0 \tag{2.1}$$

where (a, b, c, d) are the parameters to be estimated. The center of the sphere is given by $(-a/2, -b/2, -c/2)$ and the radius of the sphere $R = \sqrt{(a^2 + b^2 + c^2)/4 - d}$. The system of linear equations is solved using Singular Value Decomposition (SVD) and the parameters are estimated. Even though a small portion of the circle is used to extract the points, the

extracted points may be noisy due to the nature of the sensor, or the intrinsic calibration. Since we know the real world radius of the ball, it is possible for us to create tolerable bounds for the estimated radius of the detected sphere. By using a quick approximation, and a threshold for the radius of the sphere, we can quickly eliminate incorrect ball detection. The least square approach of fitting the sphere works very effectively, as seen in Figure 2.4.

Selection Constraint

The sphere centers are good reference points to calibrate the cameras. The sphere can be seen by any of the cameras, as long as the cameras are pointed toward a common scene. The center of the ball remains the same in the real world 3D space, no matter what camera is capturing the ball; therefore, the center of the sphere serves as an effective feature to calibrate the cameras. However, not all sphere centers can be used for representing point correspondences for calibration. Motion and spatial constraints are applied to ensure that the points selected are effective to represent the point correspondences.

Motion Constraint: The data is captured at $30fps$, the cameras are synchronized using Network Time Protocol (NTP), and are millisecond precise. Since there is no real way of regulating when an image is captured by the Kinect, fast motion of the ball can result in the deviation of the position of the center; this happens because the ball is captured at different instances of time by each camera. Using sphere centers, captured while the ball is moving fast, would result in non corresponding points being used for calibration; this would eventually lead to inaccurate calibration. So, a motion constraint is applied to ensure that the points selected remain the same (motion less than $5mm$), for at least a couple of frames, before they can be used for calibration.

Spatial Constraint: Points from different parts of the scene need to be captured to enable accurate calibration. If all the points are captured based only on the motion constraint, then a large number of points would belong to the same region of the scene. To get

representative points from the scene, and to avoid oversampling a region, a spatial constraint is enforced. The spatial constraint ensures that there is at least a minimum of δ distance between the previous point captured, and the next point captured for calibration.

2.2.2 Scene Calibration

The complete scene is calibrated in a pair-wise manner. Each camera is paired up with all the other cameras. At each camera, the point is selected using the above procedure, and is time stamped. For each pair of cameras, points within a temporal window of size ω, are collected. Once the number of points crosses the threshold η, registration is performed using the points. Subsequently, registration is performed after every η new points are added, until the transformation matrix is stable. In cases where the camera configuration allows little to no overlap between the camera views, the calibrated point pairs from neighboring cameras are used to register the two cameras.

Registration

The collected point pairs, for each pair of cameras, are registered with each other. Registering the point pairs aligns the two camera views with each other, to generate a single world coordinate system having accurate calibration. The registration problem can be stated as given, a m-dimensional point set $P = \{p_i\}$ and $Q = \{q_i\}$, where $i = 1, 2, \cdots, n$; then determine the rotation matrix $R \in \mathbb{R}^{m \times m}$, translation vector $t \in \mathbb{R}^m$ and scale c, such that the sum of the least square errors, $E(R, t, c)$, is minimized.

$$\underset{R,t,c}{\operatorname{argmin}} \; E(R, t, c) = \underset{R,t,c}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} ||p_i - (cRq_i + t)||^2 \tag{2.2}$$

There are many closed form solutions to the registration problem (Horn et al., 1988; Arun et al., 1987; Umeyama, 1991). Popular methods (Horn et al., 1988; Arun et al., 1987) are known to provide incorrect R, for highly noisy point sets (Umeyama, 1991). Even though

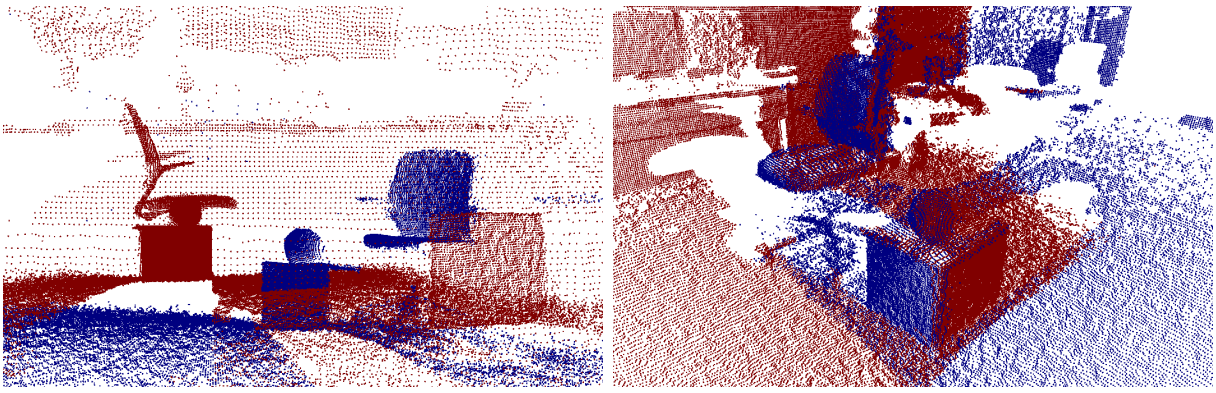

Figure 2.5. Calibration results between Kinect A (red) and Kinect B (blue), from left to right: without calibration and after calibration.

the likelihood of having corrupted point data is minimal, due to our use of multiple points to estimate the sphere center, the solution proposed by (Umeyama, 1991) is used to register the point sets. The approach first makes the points translation independent, by subtracting the centroid of the points from all the points. Then SVD is performed on the covariance matrix of the translation independent points. The reflective artifact is corrected using the determinant of unitary matrices from SVD, and the R is estimated. The scale c is estimated using the singular values. The translation t is determined by transforming the centroid to the new orientation, and calculating the difference in position. This approach provides very good alignment, even with as little as 10 good point pairs as shown in Figure 2.5.

Propagated Registration

The ball can be detected by all pairs of inwardly facing cameras; but to get accurate results, the ball needs to be placed around the scene. In large scenes, cameras located at the extreme ends have little to no overlapping region. Directly calibrating these cameras using the ball, leads to inaccurate alignments as seen in Figure 2.6, primarily due to lack of variations in the point correspondences. In situations with minimal or no overlapping region between

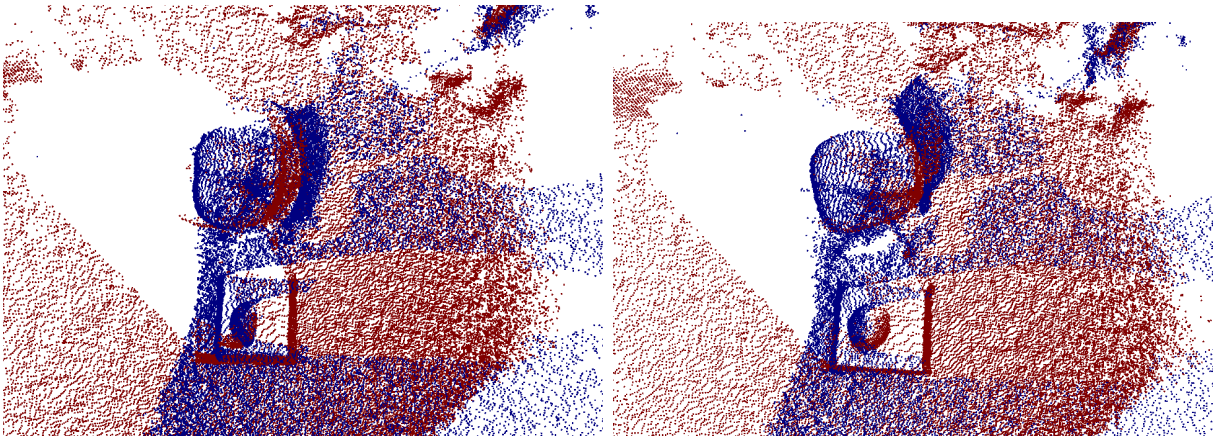

Figure 2.6. Direct registration of the points (left), and calibration by propagated registration (right), for two Kinect V2 sensors located at bottom right (red), and left middle (blue), with $5m$ of separation between them. In the scene, a ball is placed on the box next to a chair.

the cameras, it is not possible to get point correspondences with much variation. For these situations, registration between the cameras is propagated.

The exact path taken for the propagation is supplied by the user calibrating the space. For example, lets assume there are 4 cameras $(1, 2, 3, 4)$ placed $2m$ from each other in a straight line, and facing away from the line in an orthogonal direction; to calibrate cameras 1 and 4 using the path $(1, 2, 3, 4)$, the points captured between 1 and 2 are transformed to camera 4's local space, using the calibration results for (1,2), (2,3) and (3,4). The local space position of 1, and the local space position of 4, for each of the points is assumed to be a known observation, similar to the ball center. Similarly, points captured between 3 and 4 are transformed to camera 1's local space. An equal number of the transformed points are chosen from each of the paths, and the registration procedure described in Section 2.2.2 is applied to these new sets of point correspondences. If we just propagate the registration between the cameras by multiplying the transformation matrices, then the final calibration between the cameras is solely dependent on that path. Using a transformation multiplication based strategy is good when it can be clearly verified that all the cameras in the path are calibrated well with each other. The propagated registration strategy is similar to bundle

adjustment in that it tries to optimize the calibration for the distant cameras, using all the possible intermediate calibrations for the captured points. Since multiple paths are used, the quality of calibration obtained by this approach is visually accurate. Figure 2.6 shows the calibration result achieved by propagating registration through 4 different paths; the scene had 7 different cameras and the paths used all of them.

2.3 Evaluation

The quality of calibration estimated using BC was evaluated with a setup consisting of 7 Kinect V2 sensors. The sensors were arranged with a maximum of $2m$ of height variation, in a rectangular pattern, with a sensor on each side of the diagonal facing each other; please refer to the supplementary material for a sensor arrangement diagram. All the sensors were connected to an individual machine, having an Intel Xeon 3.0 ghz processor, 12GB RAM, and Nvidia Quadro 4000 graphics. The machines were connected to each other on a 1 Gbps network. The system clocks are synchronized within $\pm 10ms$ of each other, using NTP. The approach is implemented in C++, with GPU optimization on windows. The spatial constraint minimum distance δ value was set to $100mm$, to ensure a good variation as well as a quick capture. Based on the synchronized clocks, ω is set at $40ms$. η is set at 10 for fast calibration.

Ball detection is a fundamental part of the BC method, allowing the method to function without any manual intervention. To ensure that the ball is detected accurately at all times, we studied thousands of frames of tagging and noticed that about 2% of the time, the head was detected instead of the ball. The only time the head was detected instead of the ball, as shown in Figure 2.7, was when the camera was at a high elevation looking down towards the person and the scene; to avoid this problem, the person moving the ball inside the scene wore a hat or a hood.

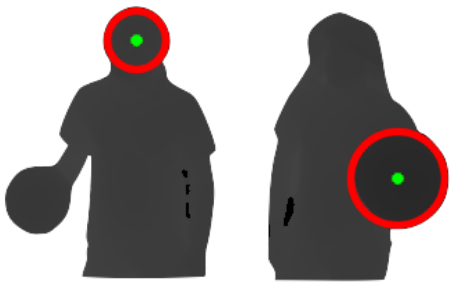

Figure 2.7. Head getting detected as ball when the Kinect V2 is placed high facing downwards (left), and wearing a hood over the head eliminates the false detection (right).

Comparison of result between BC and three prominent methods, representing each style of extrinsic calibration for Kinect sensors, was also performed. For this comparison, to avoid calibration propagation errors, only three Kinect V2 sensors were used. The sensors were positioned in the shape of an L, with each of them located at a corner; there was also some height variation between the sensors. All sensors were calibrated with respect to their nearest sensor and the result rendered as a point cloud. BC and (Kowalski et al., 2015) seem to perform the calibration well, with BC being off by at most 1cm in certain places, like on the small orange ball (Figure 2.8). The (Kowalski et al., 2015) is off near the label of the box, and other places by $2.5cm$, despite performing ICP to refine the results. The original calibration of the method was off at most places, by more than 4cm. The checker board based (Zhang, 2000) method required multiple calibration attempts, taking a significant amount of time. The best result that could be achieved by us, was off by more than 6cm at many locations, like the white Kinect boxes in Figure 2.8. The plane intersection based approach of (Auvinet et al., 2012) took a long time to calibrate, with a lot of work flipping the plane a certain way; the calibration was off by 4cm at most locations.

BC method took the least amount of time to calibrate requiring about a minute to calibrate all 3 sensors. The live scan 3D method (Kowalski et al., 2015) took about 2 to 3 minutes for all cameras. Both of the (Zhang, 2000) and (Auvinet et al., 2012) methods took

Figure 2.8. Calibration result starting from top left: our ball calibration, visual pattern approach (Kowalski et al., 2015), Zhang's method (Zhang, 2000), and plane intersection approach (Auvinet et al., 2012).

over 30mins to calibrate, with most of the time spent moving the board, or re-calibrating to get better accuracy.

Scene calibration with 7 Kinect V2 sensors in a rectangular arrangement, with sensors placed at different elevations and angles, was also performed. The entire calibration process took about 3 to 4 minutes. The sensors that were located diagonally opposite each other, were required to be calibrated by propagated registration. The scene was reconstructed fairly accurately, with certain angles having up to 2cm in error. Figure 2.9 shows the front and top views of the scene that were reconstructed using the setup.

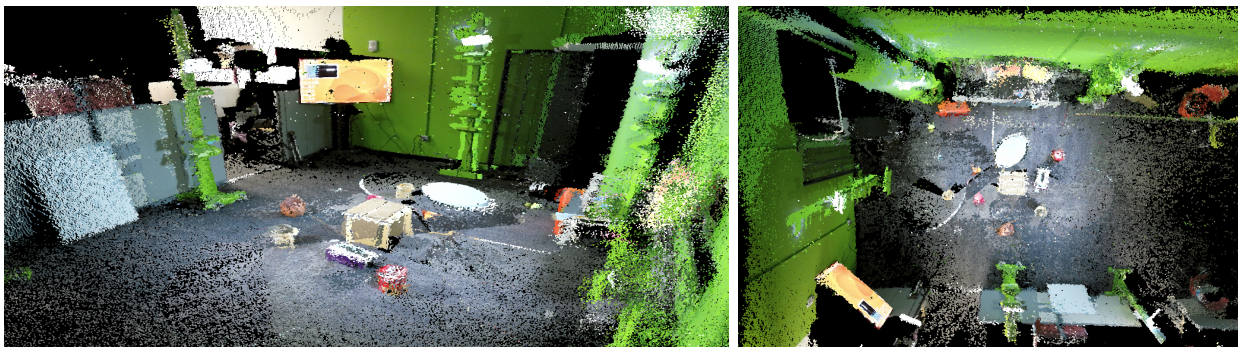

Figure 2.9. Point cloud rendering of a scene with multiple objects, captured using 7 Kinect V2 sensors, calibrated simultaneously using our ball calibration method from left to right: front and top view of scene.

2.4 Discussion

The Kinect V2 sensors use time of flight to estimate depth. When multiple Kinect V2 sensors are used to capture the same scene, interference is caused between the sensors, resulting in noisy depth estimation. For evaluating BC, 7 Kinect V2 sensors were positioned to capture a $4m \times 3m$ area, resulting in significant salt and pepper noise. Two sensors placed in close proximity, do not seem to interfere much with each other; however, for certain configurations, when the sensors face each other, there is a visually noticeable increase in noise. Applying a median filter removes most of the noise in the depth image. The Kinect V2 sensor also seems to have issues estimating the depth of glossy surfaces; so, in our method, we use an older worn out basketball to ensure accurate tracking.

There is also a considerable amount of noise generated around a plane surface, as shown in Figure 2.10; some plane based calibration methods (Auvinet et al., 2012) try to avoid this noise by fitting a plane on the surface, while others (Kowalski et al., 2015) reduce it by using filters and ignoring the rest of the noise. The amount of noise around a sphere is considerably less in the case of Kinect V2, as seen in Figure 2.10. With the use of a large number of points for estimating the center of the sphere, the noisy data does not really effect the overall accuracy of the calibration. In approaches where noise is ignored (Kowalski et al.,

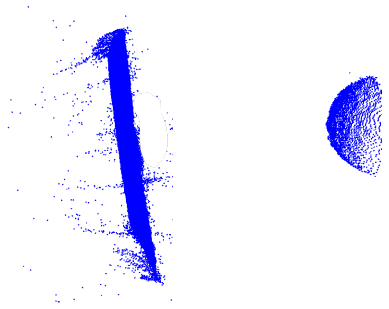

Figure 2.10. Noise while capturing a plane (left), and a sphere (right), using a single Kinect V2 sensor located on the left side.

2015), either the result is misaligned, or a step using dense point registration like ICP, is performed to improve the camera calibration.

The calibration between the color and depth cameras of the Kinect V2 sensor is not very accurate, mainly due to the elliptical conics of the time of flight depth sensor; as a result, most of the images bleed color into the object next to them. This same issue is also noticed in all the methods that rely on color image, to register points or calibrate the sensors. By just using the depth image, BC achieves better results. The Kinect V2 sensor accurately measures depth in the range of $0.5m$ to $3m$ (Yang et al., 2015); after $3m$, the depth accuracy degrades significantly. We avoid positioning the ball beyond $4m$ from the camera, to ensure reasonably high accuracy of the point used for calibration.

CHAPTER 3

RENDERING

The 3DTI systems capture the user, using multiple cameras, to render a realistic 3D model of the user in the virtual world. A new 3D model is reconstructed and rendered, every frame, to ensure a one-to-one correspondence between the user's actions, in the real and virtual world. In order to achieve better performance, instead of creating one 3D model for the user, our approach creates a 3D model for each captured image of the user. Each of the different meshes are rendered together to give an impression of a single watertight 3D model of the user.

Each camera machine processes the scene captured by each camera, independent of each other, as shown in Figure 3.1. Each RGB-D camera provides the color and depth image of the entire scene. The noisy images are filtered to improve their quality. The user is then segmented out of the captured image. The segmented user data is then projected to 3D, using the camera's intrinsic parameters. This results in a point cloud representation of the user. The point cloud represents each pixel in the image as a 3D point, so depending on the resolution of the depth image, the resulting point cloud may be extremely dense and look like a complete structure, or be sparse, with no clear visible structure. A mesh is created from the point cloud to ensure a good quality rendering of the user, irrespective of the depth image resolution, or the angle of rendering. The texture of the mesh is rendered by using the color image captured by the camera. The color image captured by the camera, is mapped onto the vertices of the mesh, using the extrinsic calibration of the color and depth cameras.

While rendering overlapping meshes, as is the case in situations involving multiple cameras, alpha blending shader is used to avoid mesh overlap artifacts. The shader also ensures the smooth rendering of shadows for the user's model. The adaptation to light and the smooth rendering, ensures that the user's mesh is immersed completely inside of the virtual world.

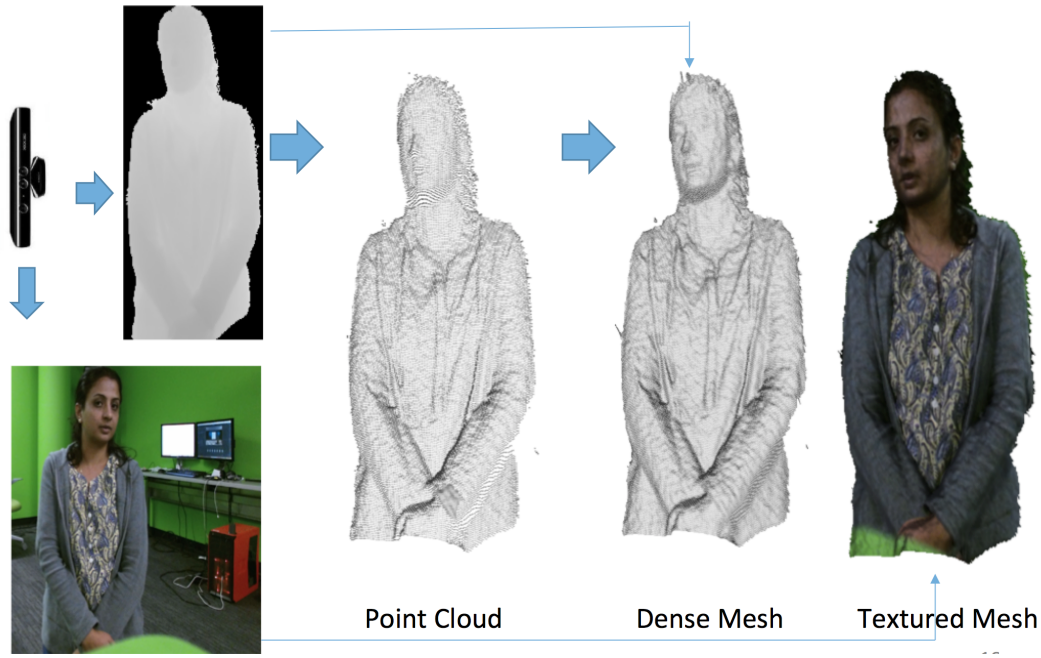

Figure 3.1. The processing pipeline for each of the cameras, in an i3DTI system.

3.1 Related Work

There are many approaches to reconstruct a 3D mesh from structured and unstructured point clouds. In this section we focus on real time approaches that are or can be used for 3DTI.

3DTI data transmission: A number techniques have been proposed in literature to efficiently transmit 3D tele-immersion data. (Yang et al., 2006) presented a multi-stream adaption frame work for 3DTI having two major steps - view based stream selection and bandwidth dependent content adaption. A major issue associated with this approach is that pixels are selected evenly without giving importance to fine details. (Shi et al., 2009) proposes view dependent 3D video compression technique for mobile devices. (Redert et al., 2002) and (Yang et al., 2006) use image and zlib compression, respectively. (Chen and Nahrstedt, 2013) proposes activity aware adaptive compression by combining activity recognition and real-time morphing based compression. (Kum and Mayer-Patel, 2005) explores inter stream

redundancy to achieve real time depth stream compression in 3DTI scenario. A model based approach proposed in (Raghuraman et al., 2013) extracts kinematic parameters of human body as skeleton information, which is further used for motion estimation followed by compression. (Kurillo and Bajcsy, 2013) uses bisection algorithm meshing proposed in (Vasudevan et al., 2011) based on depth variation, and performs efficient depth data encoding and compression using this mesh hierarchy. In (Wu et al., 2011), authors perform a psycho-physical study to come up with perceptual threshold for color-plus-depth level of details which is used to correlate it with depth variation threshold in the bisection algorithm meshing to come up with adaptation scheme for resource management.

Our survey indicates that very few techniques in literature focus on designing bandwidth adaptive 3DTI system. Most of these methods are based on the key concept of reducing the data size in order to increase *fps*. In this paper, we address the challenge from a different perspective, by designing a bandwidth adaptive system that jointly determines optimal size of 3D and texture data for available bandwidth. We also define the notion of quality used in selection criterion, which allows us to apply different level of details in different parts of body.

Meshing: Many methods are available for performing triangulation of 3D data, most of which take a significantly high amount of time leading to non real-time rendering. Image based meshing is faster, (Vasudevan et al., 2011) proposes a bisection algorithm for triangulation based on amount of variation on gray scale values. It uses the fact that every point in triangular region should be at same depth. Hence, if there is significant variation in depth values inside any triangle then it should be further bisected to create smaller triangles. (Raghuraman et al., 2013) proposes image based meshing technique to generate a fast dense mesh which uses the underlying gird structure of depth image. However, when key vertices are selected from depth image, underlying grid structure is lost. In order to address this issue, we use fast image based sweep-line constrained delanuay triangulation (Domiter and Zalik, 2008) along with segmentation information.

Figure 3.2. An orthogonal rendering of a point cloud of a plane, captured using the Kinect V2 without any noise filtering (left), and the results of the use of median filtering (right).

3.2 Noise Removal

RGB-D cameras capture the color and depth images of a scene. The color image is relatively noise-free and has a higher resolution. The depth image, depending on the technique used to generate it, suffers from different kinds of noises. The two popular techniques for generating depth images are structured light and time of flight.

The structured light based depth cameras, like the Kinect V1, consists of an infrared (IR) projector and an IR camera. The IR projector projects a fix set of patterns, which are captured by the IR camera. Based on the differences between the dots in the projected and captured pattern, the depth of the various points in the 3D world are estimated. This technique leads to holes, especially around the edges, and in general, a lot of edge noise. The noise can be reduced significantly by applying a bilateral filter on the depth image.

Time of flight based cameras, like the Kinect V2, construct the depth image by emitting a ray of light, and measuring the time it took for the light to get captured. This technique leads to a lot of salt and pepper noise, especially around the edges. This noise can be minimized substantially by using a median filter. The raw noise, during the capture of a plane facing the Kinect V2 camera, and the corresponding median filtered result, with reduced noise, are both shown in Figure 3.2.

3.3 Segmentation

The cameras capture the entire scene, including both the user and the background. Segmentation is the process of extracting the relevant foreground object from the captured images. For the purposes of i3DTI, only the user needs to be reconstructed, so the background needs to be segmented out. In situations where other objects need to be reconstructed, like the ball when performing calibration, other specific segmentation approaches need to be used. There are many techniques for segmenting a foreground object from a color or depth image. In this section, our primary focus is only on those techniques which can extract the user from the scene in real-time.

3.3.1 Person Extraction

The Kinect provides the skeleton information of the user, along with the color and depth images. The user can be extracted from the scene, by overlaying the skeleton on the depth image and using a region growing approach to identify the silhouette of the user, from the depth image. Irrespective of the level of accuracy of the skeleton, the hip center joint of the skeleton always corresponds to a point on the user's depth image. So, the hip center is used as the origin for the region growing approach. The region continues to grow in all directions, away from the hip center, as long as the variation in depth values is within a threshold. The threshold is determined based on the resolution of the depth image, and is typically about $10cm$ for a Kinect V2 image. Both the captured scene and the extracted user, using the region growing approach, are shown in Figure 3.3.

3.3.2 Volumetric Segmentation

Sometimes the capture area can contain many different objects, along with the user. It is possible, under these circumstances, that either the Kinect does not detect the user at all,

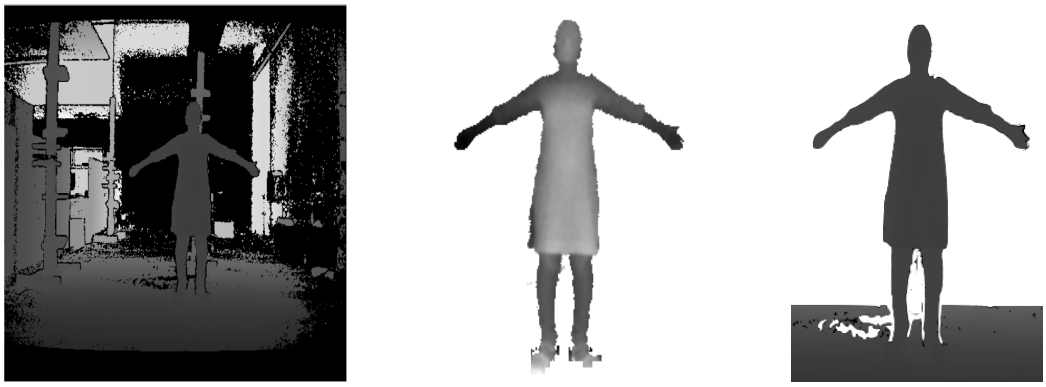

Figure 3.3. Segmentation from left to right: captured scene, person extracted using region growing and volumetric segmented scene.

or returns an inaccurate skeleton. To extract the user appropriately, from these types of scenes, volumetric segmentation is used. The volumetric segmentation approach relies on the fixed camera view, to create a 3D background representation. When the user enters the scene, there is a clear change in the 3D representation of the scene. The 3D background representation of the scene is overlaid on the current 3D representation of the scene, and all of the points within a certain distance, from the background point, are removed in the newer scene. The extent of background removal, depends on the distance threshold used. A greater distance threshold would be able to completely remove the floor, to extract only the user, as shown in Figure 3.3.

3.4 Image Meshing

Reconstructing a surface, from unorganized 3D point clouds, is a challenging problem that is processing intensive. Most 3DTI techniques represent the depth information as a point cloud, before generating a 3D mesh. This causes the reconstruction pipeline of these 3DTI systems to be slow, and require lots of system resources. The main reason for the high resource utilization and slower performance is the lack of neighborhood information in the point cloud representation.

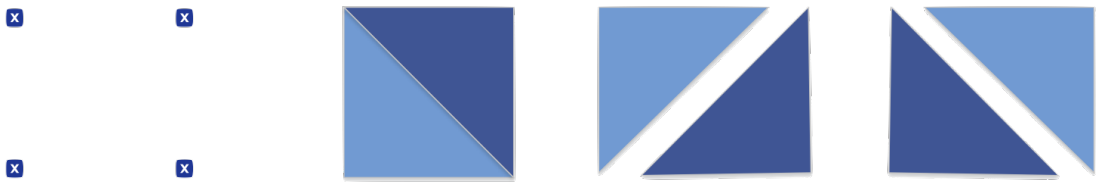

Figure 3.4. The square pattern used for triangulation from left to right: the selected points, the 2 triangles if all points are within range, and possible triangle options depending on the proximity of the points.

The depth image implicitly maintains the neighborhood information for each of the pixels, within the structure of the image. The image meshing approach exploits the clear availability of the neighborhood information in order to create triangles, quickly and effectively. This approach uses a square pattern to identify the triangulations required for the entire image. 4 adjacent pixels, in a square pattern, are selected to form triangles. For a set of four points, as shown in Figure 3.4, only 4 different combinations of triangles can be formed. A distance measure is used to determine if a triangle should be formed, between all 4, or any of the 3 points, that are selected. If the points that are selected are within a distance of the meshing threshold, then a triangle is formed between them, and the next set of 4 points are selected. If all 4 points are within the meshing threshold, then 2 triangles are used to represent the region. The approach processes each and every pixel, selecting the left, top left, and top, neighbors of the pixel, to create the square structure that is used to evaluate the feasibility of triangulation.

Since each pixel is processed, using the square representation, it is possible to implement the entire meshing technique, parallelly on the GPU. In the GPU implementation, each pixel is assigned to an individual GPU core, allowing the entire mesh to be generated in a few microseconds. The sequential CPU implementation of the approach is also capable of generating the mesh in a couple of milliseconds. The use of the image meshing technique ensures that the i3DTI system is able to reconstruct and render the user efficiently allowing highly responsive and detailed interactions.

3.5 Low Poly Reconstruction

The image meshing approach relies on the compact arrangement of points in the image to generate a mesh representation. Depending on the size of the depth image and the number of pixels associated with the object, the vertex and triangle count of the generated mesh can vary. Large meshes need more resources, transmission time and processing to render, so if the mesh generated is large then the rendering pipeline would take longer to render the mesh on screen than a mesh with lower polygon count. The compact structure of the depth image representation can again be exploited to create lower polygon mesh representations of the captured object.

The simplest way to reduce the number of vertices and triangles in the meshes that are generated from the depth image is to reduce the size of the depth image itself. A depth image scaled down to half the size would lead to about four times reduction in the number of vertices and triangles in the resulting mesh. So for a uniform reduction in the quality and the size of the mesh, the depth image is reduced in size using Gaussian pyramidal reduction. An example of the mesh created using the original depth image and the reduced depth image is shown in the Figure 3.5. The reduction in the size of the depth image also leads to up to a four times faster meshing on the CPU.

3.6 Sparse Meshing

The low poly meshing approach generates meshes of lower resolution, by uniformly degrading the denser mesh. While this approach is useful for creating lower poly meshes very quickly, the overall quality of the returned mesh is much lower than the original dense mesh. Instead of reducing the vertices and triangles uniformly throughout the mesh, it would be far more beneficial to reduce the vertices/triangles from areas having little detail (arms, abdomen, etc,), and retaining more vertices/triangles in regions with greater detail, like the face.

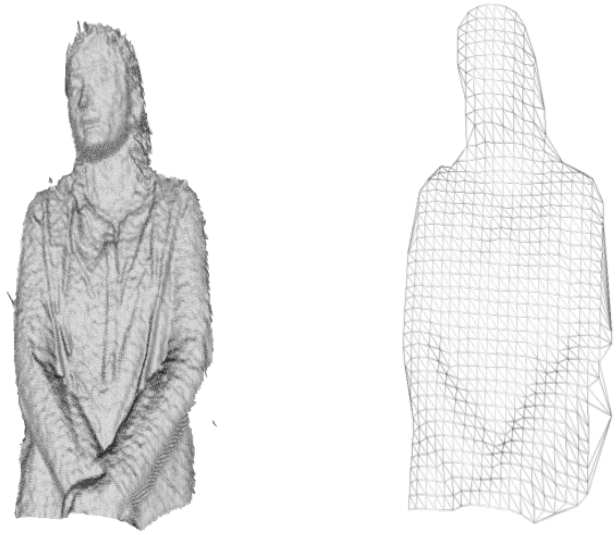

Figure 3.5. Meshing results: High resolution dense mesh created using the original depth image (left), and the lower resolution uniformly sparse mesh generated from a Gaussian scaled down depth image (right).

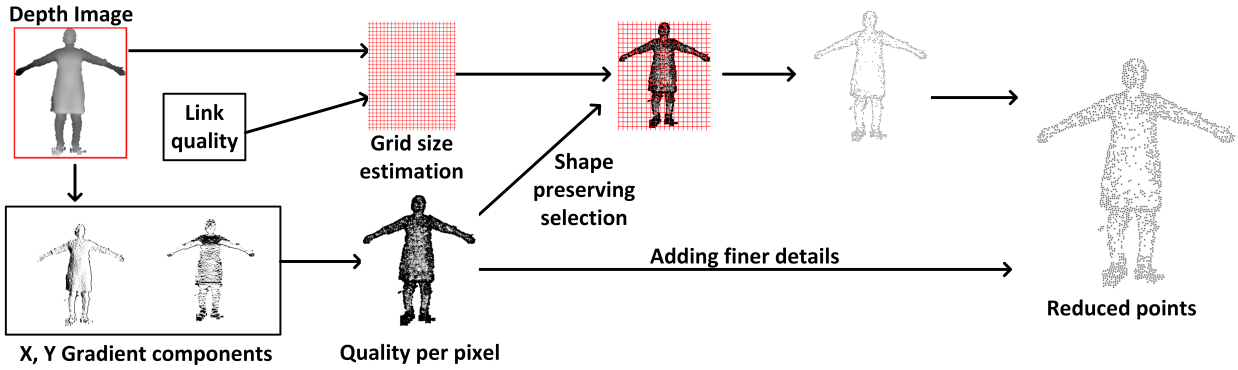

Figure 3.6. The steps involved in the sparse mesh generation approach.

The quality preserving sparse mesh generation approach uses the curvature of a region to determine if that region should have more, or less, triangles. This problem is extremely time consuming when performed in the 3D space; so similarly, as with the low poly approach, the sparse mesh approach is also applied to the depth image. The sparse mesh is generated from the depth image, as shown in Figure 3.6.

The importance of each pixel, in the depth image, is established by the curvature associated with the pixel, that is calculated using the X and Y gradient components of the depth

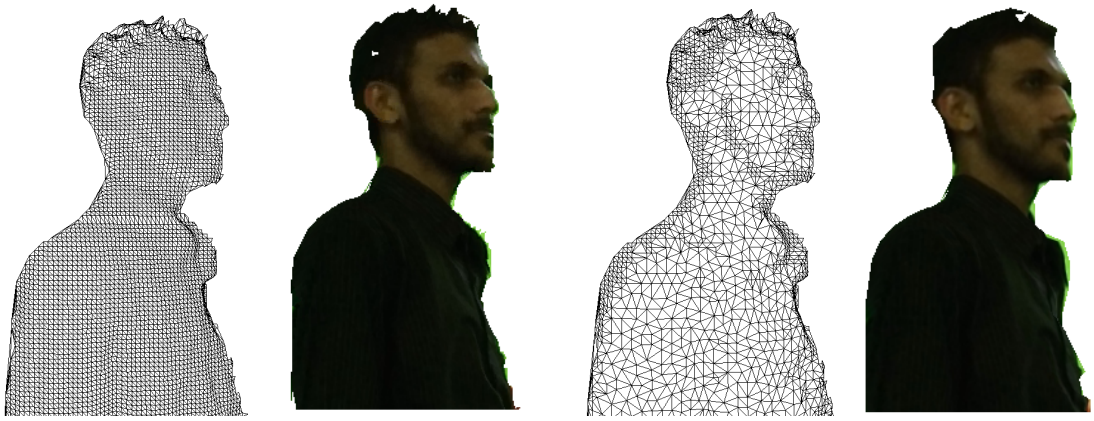

Figure 3.7. The sparse meshing of the profile of a user, from left to right, 12,000 vertices mesh, textured mesh, 1,000 vertices mesh, and textured mesh.

image. The per pixel quality measure of the depth image provides a reasonable estimate for the importance of the pixel in a local region. To get the high level structural aspects from the depth image, a sparse grid structure is used. The depth image is divided into cells, by a sparse grid, and a pixel, with the highest quality rating, is selected in each of the cells. The highest quality rated pixels are selected as the vertices. The selected vertices are then meshed together using a sweep-line constrained Delanuay triangulation (Domiter and Zalik, 2008). The result of the sparse mesh approach, for the profile of a captured user, is shown in Figure 3.7.

3.7 View Dependent Rendering

The camera arrangement in our setup allows a larger playing area, resulting in only a subset of the cameras capturing the user in the scene. Merging all the camera meshes every frame to create one single mesh for rendering is time consuming. Rendering textured range images captured from different camera poses is prone to texture merging artifacts (Pulli et al., 1997). To achieve better display and faster rendering, we use a View dependent rendering approach similar to (Pulli et al., 1997).

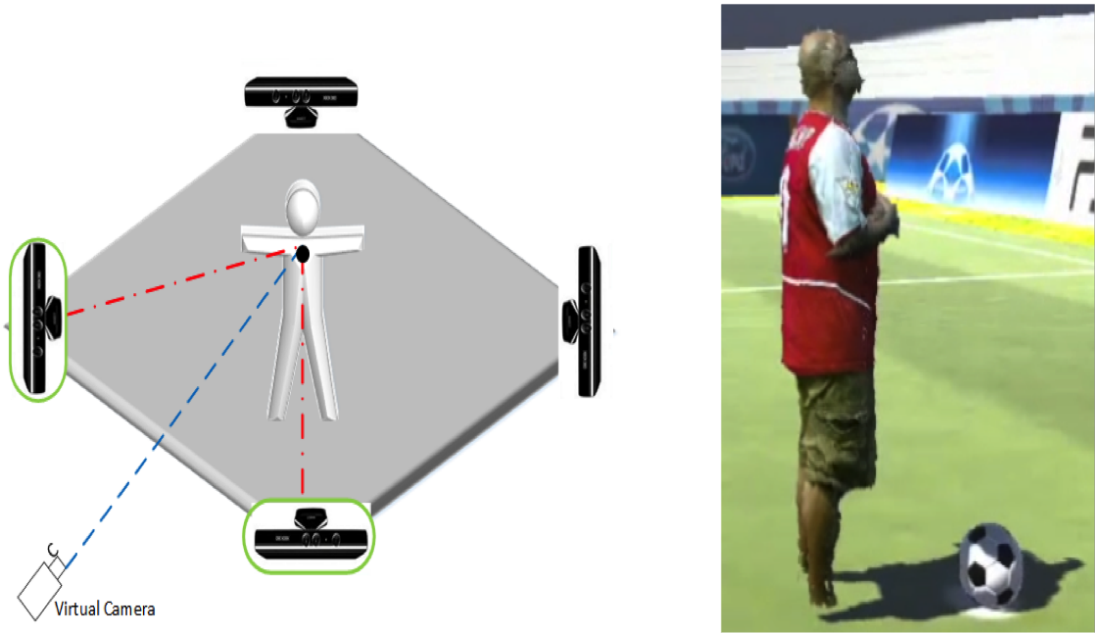

Figure 3.8. Left: View dependent rendering based on the position of the virtual and real world camera. Right: A user model rendered in the scene using view dependent rendering.

As the name suggests the meshes that are transmitted and rendered are dependent on the viewing angle of the virtual camera. Camera meshes are selected based on the similarity between the viewing angle and the capture angle of the object being rendered. If the object is outside the field of view of the camera, none of the meshes are rendered. As shown in Figure 3.8, if the angle of the ray cast from the virtual camera to the point on the object and the ray cast from the capturing camera is less than a threshold, then the mesh from the capturing camera is rendered. The threshold is determined based on the location and angles of the cameras in the capture area. For faster approximation, the shoulder center joint of the skeleton returned by all the cameras is used to determine the position of the user in the scene. This joint position is taken as the point on the object and is used for determining the meshes to be rendered. The meshes are rendered as shown in Figure 3.8 using an alpha blending shader that colors the texture based on the vertex normal, captured camera angle, and viewing camera angle.

Shadows give the user a sense of 3D perception inside the game. As seen in previous studies (Miles et al., 2012), using shadows increases the tracking and positioning capabilities of subjects in virtual environments. The same machine is used for both rendering and capturing the front view of the person. Since the processing is mostly done in the CPU without any transmission involved, the front view of the person is used to generate the shadow of the player.

3.8 Rendering Overlapping Meshes

The rendering approach creates a different mesh for each of the camera views. To avoid disruptions in the capture of the object, typically the scene cameras are arranged to have overlapping field of view. This common field of view results in the captured data containing significant amount of overlap that is converted and carried over to the mesh. If all the meshes are rendered directly then the result using a standard fragment shader is undiscernible as shown in Figure 3.9.

The bad quality rendering of the overlapping meshes is primarily due to the lack of specific ordering or prioritization allotted to the various surfaces of the meshes. To achieve better quality rendering surfaces from each of the meshes need to be selected and prioritized appropriately.

An Alpha Blending Shader (ABS) approach implemented at the fragment shader level is used to provide blend weights for each of points of the meshes to create good quality rendering. ABS uses the principle axes of the real world cameras, the mesh surface normal and the principle axis of the virtual camera to estimate the alpha blending weights for each of the mesh fragments. For a setup with n cameras each generating meshes $(M_1, M_2, ...M_n)$ every frame, the alpha blending weight for a surface j on mesh i with normal S_j^i is given by

$$A_j^i = \frac{(P_i.P_v)(P_i.S_j^i)}{\sum_{k=0}^{k<n}(P_k.P_v)(P_k.S_j^i)} \tag{3.1}$$

Figure 3.9. The effect of rendering overlapping meshes without an alpha blended shader (left), and with a specialized alpha blended texture shader (right).

Where P_v is the principle axis of the virtual world camera and $(P_1, P_2, ...P_n)$ are principle axes for the capture cameras. The result obtained by ABS as shown in Figure 3.9 is far superior compared to direct rendering and is similar to the real world capture of the person. The ABS rendering creates an illusion of a single unified mesh of the user even though none is made. Using sufficient number of cameras the ABS approach can lead to a clear 360° rendering of the user that looks exactly like a watertight 3D user mesh.

CHAPTER 4

INTERACTION

An i3DTI system reproduces the user as a "live" avatar in the virtual world. To create this 3D model, the user is captured using multiple RGB-D cameras. Since the user is reconstructed "as is", any external input devices would also be captured, and reconstructed in the scene. For this reason, to make the visual appearance of the user more consistent, the use of external devices should be avoided as much as possible.

The use of external input devices interferes with the immersive nature of the i3DTI system, by introducing possibly more complicated ways of interacting with the system. Creating a more natural exchange with the system, will increase the user's level of interest, involvement, and ability to interact with the system, thereby eliminating the need of any external devices. Using the user's pose and gestures to model all of the communication inside of the system, the i3DTI system create interactions that are both visually appropriate, and naturally consistent. In this chapter, multiple methods are presented to capture the user's actions, by accurately estimating their pose. Interaction techniques, that model these detected poses inside the virtual environment, are also presented. The combination of these approaches, provides the user with intuitive ways of utilizing the system.

4.1 Skeleton Joint based Interactions

The rendering techniques represent the real world person in the virtual world, by showing the "live" 3D model of the person. Similarly, the skeleton of the person serves as their interaction model in the virtual world. The skeleton consists of multiple bones and joints, each joined to an associated position in the 3D world. Objects can be assigned to these joint positions and moved appropriately, based on the motion of the joint. For example, a helmet can be placed on the user's head, and moved around as the user moves within the scene.

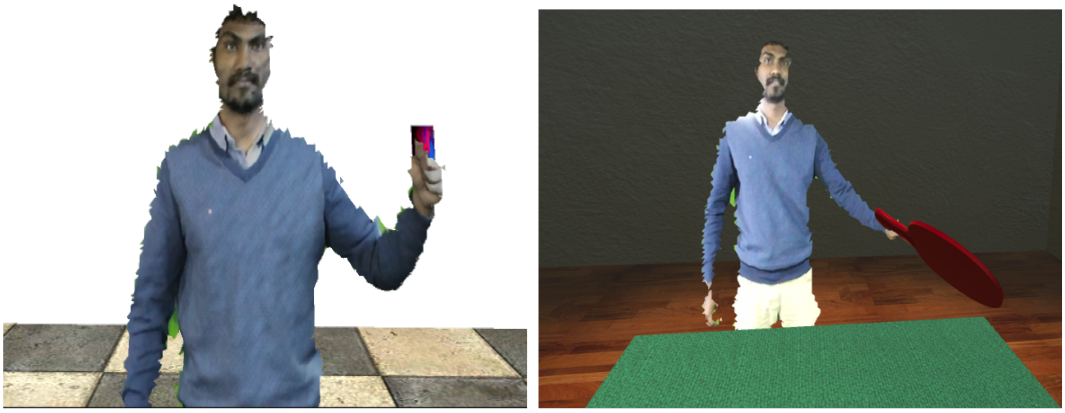

Figure 4.1. Joint based interaction from left to right: Picking up objects by closing the hand, and positioning the table tennis racket in the hand.

Joints themselves, do not provide any orientation information, so a direct association of the virtual world objects to the user's joints, would lead to unrealistic object orientations.

Skeleton bones connect two joints together, and has both an associated position and orientation. So, while the object can be positioned at a joint, the orientation of the object can be determined based on the orientation of the bone. Using the same example as above, instead of the helmet moving with the head joint, the helmet can be rotated based on the orientation of the neck-to-head bone; this provides a much more realistic looking integration of the virtual object, with the 3D model of the person.

For certain joints, like the hand, that consist of smaller bone regions (fingers), extra pieces of information can be identified. It is possible to track the open/closed hands of a person. Using the tracking, interactions that enable the user to pick up or release objects, using their hands, can be created. Figure4.1 shows how a mug can be picked up from the table, just by closing the user's hand.

In order to obtain a reasonably accurate orientation of the hand, the hand needs to be open. So, in situations where a racket needs to be controlled by a hand, a slightly

unrealistic looking modeling is performed, as shown in Figure 4.1. Despite the awkward visual representation, the gameplay is greatly improved by using open hands to move and rotate the virtual racket.

4.2 Full Body Interaction

The joint based interactions allow users to pick up, and manipulate objects by using their terminal joints, like their hand, head, feet, etc. These joint interactions are largely limited to the motion of the virtual object, and are based on the corresponding motion of the joint/bone. A user does not always need to use specific pick up/release motions, in order to physically move objects in the real world. Cause/effect based interactions are also possible; an example of this would be, how walking through grass causes the grass to bend. The full body interaction technique enables the virtual environment to react, based on the user's entire body; this is achieved by creating a collision model around the physical body of the user, in the virtual world.

The i3DTI system reconstructs a new full 3D model of the user, every frame. The reconstruction process is resource intensive for each camera, and requires lots of data transmission, in case a complete model of the user needs to be created. The use of view based rendering, and other partial mesh optimization approaches, improve system performance, while simultaneously reducing resource utilization. The i3DTI system reconstructs a mesh, for each camera view, to ensure faster performance. The view based rendering system never generates a single mesh, to represent the person inside the scene. Since it uses view based streaming, the rendering machine does not even contain a complete set of meshes that represent the person. Even if all of the views were present, creating a mesh based collider every frame, is too computationally intensive to be feasible for a real-time low delay system. So, instead of generating the collision model based on the mesh of the person, our system relies on a primitive box and capsule colliders to cover the entire human model. Box colliders map

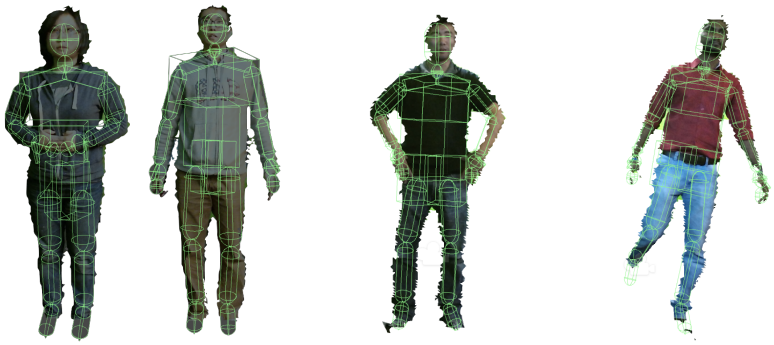

Figure 4.2. Skeleton based full body colliders for various users.

the chest and abdominal region of the person, while all of the other parts, including the head, are represented by a capsule collider.

The orientation and scale of each collider is determined by studying the skeleton of the person. Since the skeleton represents the medial of the person's mesh, it is possible to extract a reasonably good estimate of the size of various parts of the body, using just the skeleton. The human body is largely symmetrical, and many parts of the body maintain proportions to each other. Various parts of the arms and legs are proportional to each other, while the size of the thighs are relative to the hips, as are the biceps to the chest. Using these associations, we built a model to empirically estimate the coefficients, used for determining the size of the limbs, given the joint positions. We captured both the mesh and skeleton of 8 individuals (6 male, 2 female), of differing statures, in different poses. The meshes were then segmented, and the colliders were fit, to get a scale estimation for each of the segments. As expected, the length of the colliders for the limbs was found to be the same as the skeleton limbs. However, the scale of the terminal colliders, like the head, feet, hands, etc., are almost double the length of the bone. Weights were estimated, to map the position of the joints to the width and radius information of the colliders, for each of the bones. Using these weights for any given skeleton, our system generates the collider estimation, as shown in Figure 4.2. Although not obtained from a statistical study on a huge population, the colliders have been accurate for any of the users using the system.

Once the scale of the colliders is determined, the orientations of the skeleton bones are applied on the colliders, to accurately map the human being rendered. The entire human model is made collision capable in microseconds, without any mesh or point cloud information, by using the proportionality weight strategy for the collider scale estimation.

4.3 Multiple Kinect Pose Detection

Human pose identification has various applications in activity recognition, natural user interface, gate analysis, etc. RGB-D cameras, with their extra depth stream, allow us to get an accurate estimation of the person's pose in real time. However, these skeleton estimations can quickly become inaccurate, mainly due to occlusion. The use of multiple cameras, to capture the same scene, allows us to obtain more information about the person and their pose. Current approaches, used to identify the skeleton pose using multiple cameras, either take too long (Shuai et al., 2017), use extra sensors (Wu et al., 2014), or require millions of training examples (Shotton et al., 2013).

Even multiple Kinect based approaches (Yeung et al., 2013), use a small number of Kinects to capture the user in a small space, with the person facing a fixed direction. Using a large capture space with multiple Kinect sensors, to allow complete freedom to move around in the space while tracking the user's pose, is not normally explored because of the issues that are caused: partial depth images of the person, fast motion, no user direction information, etc. To allow a skeleton to be estimated in real time, no matter where the person is in the activity space, we propose Joint Accuracy based Consensus (JAC).

JAC utilizes the skeletons and depth images from all of the Kinects, using the view of the person to estimate an accurate skeleton. It assumes that the individual joint estimations, from the Kinect, are more reliable than the complete skeleton. All of the joints are evaluated to determine the Probability of an Accurate Joint (PAJ) being estimated precisely. Then, a distance constrained greedy consensus approach is used to select the joints, such that the

joint, PAJ, of the set is the maximum. The direction of the person is then used to determine the right side of the body, from the left side.

We determine the direction of the person using the feet, knee angle, or torso and head, to ensure that all of the joints are labeled correctly, irrespective of which direction the person is facing. While it is possible to use face tracking to determine the direction that a person is looking in, it is not necessarily the same direction in which that person's body is aimed, or positioned.

We developed two methods to measure the accuracy of the skeleton, by using all of the views captured by the different Kinects. The containment score measures the error in the skeleton by ensuring that the skeleton is always inside the person, irrespective of the camera viewing angle, or the pose. The coverage score determines the error, in the extent to which each skeleton bone covers a part of the body; it measures the error by verifying the amount of points associated with each bone, after Voronoi segmentation.

4.3.1 Joint Accuracy based Consensus

A Kinect sensor provides the color, depth, and skeleton estimates for the captured scene. The skeleton estimate is generated based on just the depth image, captured by that particular sensor. The sensor captures the depth information using a time of flight approach, which also results in large occlusions for any poses where the person is not directly facing the camera. Since the Kinect was designed for a more natural user interface, it assumes that users will be facing the Kinect, while performing all of their activities. It uses a depth variation based interest point selector, and descriptor, to allow the tagging of the individual parts of the depth image. A random forest based classifier is trained with millions of real and synthetic depth images, of different sized users, performing various activities. Using these classifiers, various parts of the body are tagged, and then connected, to generate the final human skeleton. We believe that the Kinect sensor approach, though proprietary, has a very high

accuracy, especially in detecting the location of the parts of the body, during non occluded poses; this is mainly attributed to the huge amount of training data that is used to train the classifier. The implementation of the classifier on the GPU, allows skeleton estimation in real time; allowing us to estimate (Raghuraman et al., 2013), animate (Raghuraman et al., 2015), or even interact (Raghuraman et al., 2012) with, the 3DTI system more naturally. So instead of capturing a large training data set, with multiple depth cameras, we leverage our approach on the accuracy of the Kinect's body part detection.

Our Joint Accuracy based Consensus (JAC) approach uses the depth and skeleton information, from multiple Kinects, to generate a single skeleton of the person in the scene. Instead of relying on the entire skeleton, which might be prone to errors caused by various factors (Section 4.3.2), we treat the skeleton joints independent of each other. Treating these joints separately, ensures that our method uses only the primary classification results of the Kinect approach, as described in (Shotton et al., 2013). We then calculate the probability of an accurate joint estimation by the Kinect, based on the joint position in the local Kinect depth image, and the trends observed during experimentation. Even though the Kinect provides joint state information, indicating whether the joint is tracked, inferred, or not tracked, we found this information to be very unreliable in our experiment; because of this, we decided to use only the joints whose joint state is tracked by the Kinect.

After calculating the PAJ of these tracked joints, we transform all of the joints from the various Kinects to a single 3D space, and then re-estimate the side of the joints, based on the location of their origin in the 3D space. Once all of the joint sides are re-estimated, we use a distance constrained probability maximizing the consensus approach to determine the location of the joints in 3D space. The direction of the person is determined by using the legs (whenever possible), or the upper body, if no reliable leg information is available. Using this direction, we tag the joints appropriately to their corresponding sides. The entire flow of JAC is shown in Figure 4.3.

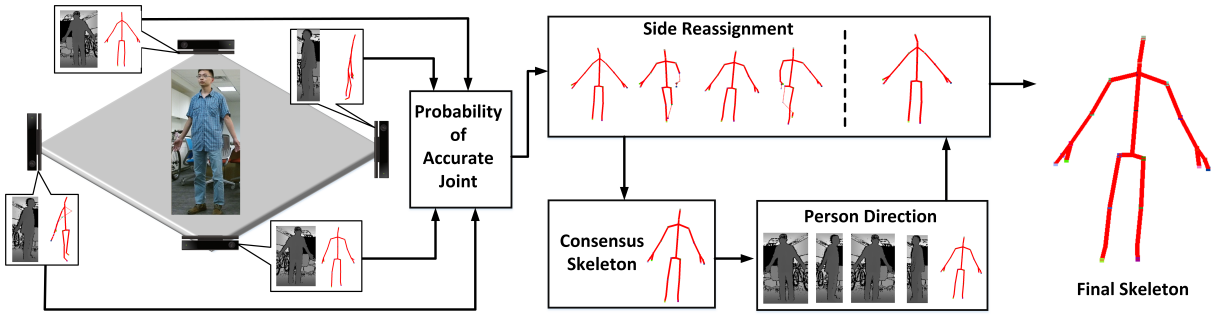

Figure 4.3. Overview of our skeleton estimation approach.

Even though the accuracy of the Kinect joints is majorly affected by occlusion, most of these issues can be resolved by using multiple cameras. Other factors also need to be considered; in the next section, we elaborate on these factors and explain our approach in detail.

4.3.2 Kinect Skeleton Defects

The single frame single camera skeleton identification of the Kinect is largely accurate; but in certain situations, it is vulnerable to many problems. In order to better understand the issues, we group them based on the source of the issue, as follows:

Sensor Noise: Kinect V2 sensors use the time of flight to measure the depth at each pixel. This measurement is influenced by lighting, the reflectiveness of an object, dust, curvature, etc. The depth sensor has a relatively low resolution of $512x424$ pixels, with a wide angle lens; so, depending on the location of the person (at least 1m from the camera is needed to capture an entire person), a relatively low number of pixels are used to encode the person, causing regions like the hand and fingers, to be captured with lower level detail.

Occlusion: For accurate skeleton identification, a direct line of sight is necessary between the camera and the different parts of the body. These line of sight problems, or occlusion, can be categorized as either external, or self occlusion. External occlusion occurs when an external object occludes the object that is being tracked. Self occlusion occurs when

deformation, or the pose of an object, hides certain other parts of that object; for example, when a person stands with their hand behind their back, the back blocks the view of the hand.

Object Interference: The position of objects, even if not occluding the view of the camera, plays a key role in the detected skeleton; for example, if a person is sitting in a chair, and the detected skeleton incorporates the chair as part of the person, this results in an inaccurate skeleton. Even in an area with no external objects, the ground interferes with the detection of the leg, especially the knee and the foot.

Motion Blur: When actions are performed very quickly, there is a noticeable difference in the alignment between the depth and skeleton data. This particular problem might be caused by a difference in the frame that was used for skeleton estimation compared to the depth information that was captured, or it could just be from sensor noise.

Fitting: The skeleton is an estimate of the pose of the person, and should ideally be located in the middle of the body. Since only one camera view is available, the skeleton may be too close to the surface, or too far behind the person. It was noticed that, Kinect detected skeletons are often times too far away from the surface of the body.

Lighting: The lighting, and the reflectiveness of the surfaces surrounding the person, have significant influences on the overall skeleton that is detected. The Kinect sensor has issues in estimating the depth of shiny black surfaces; because of this, the location of the head and shoulder region is prone to errors, when capturing the person from behind.

While some issues like object interference, lighting, and sensor noise (to some extent), can be reduced by managing the capture area appropriately, other issues, like motion blur and edge noise from a sensor, can only be resolved with the use of better sensors, or sensor specific filtering approaches. Fitting errors are mainly caused by partial viewing angles and occlusion. In our 3DTI setup, we reduce depth image errors by using non-reflective carpet and controlled lighting. A set of calibrated Kinects was used to reduce the level of occlusion significantly, for a large number of poses.

4.3.3 Probability of Accurate Joint

When each Kinect only has a partial view of the person, it is highly likely that none of the Kinects will produce a complete and accurate skeleton. Picking a skeleton based on the point of view, or other factors, is not feasible. Treating the skeleton as a hierarchical set of joints, allows us to use all of the estimated skeletons appropriately, to form a single accurate skeleton of the person. All of the joints are considered, independent of each other, but are then ordered based on hierarchy, allowing the root joint to be processed first, and leaving the the terminal joints to be processed at the end. Various independent factors influence the accuracy of the joint directly. Each of these factors can be equated to a probability, which in turn, can combine together to provide the Probability of an Accurate Joint (PAJ). The key factors, that are referred to above, are listed below:

Occlusion The major cause of skeleton inaccuracies is occlusion; the use of multiple cameras reduces occlusion significantly, but the problem still persists in the skeletons estimated from each Kinect. It is obvious that, when parts of the body are not clearly visible to the Kinect, the corresponding joints are interpolated at best. So, if we say $P(O)$ is the probability of occlusion, then $P(\overline{O})$ determines the PAJ. We can determine the likelihood that a joint is occluded, by segmenting the entire depth image using the skeleton. We use a region growing based approach seeded at each joint, and the corresponding bone. When two regions meet, the segment with the minimum distance of the point from the bone in 3D, is assigned to the point. If the parent bone has no points in its segment, then that joint is definitely occluded. Also, if the immediate neighborhood of the joint does not contain any of points from its segments, then the joint is occluded.

Joint Position The depth estimations of a Kinect sensor are accurate only in a fixed region (Yang et al., 2015); so, if the joint exists outside this region, the probability that the joint is accurately estimated diminishes, the more it continues moving away from the region. So, the PAJ based on position, $P(L) = (|\tan^{-1}\frac{x}{z}| - \tau_x)(|\tan^{-1}\frac{x}{z}| - \tau_y)(z - \tau_z)$, where τ is

the focal point of the depth camera. **Bone Angle** The angle of the bone, in correlation to the capture plane of the camera, plays a vital role in determining the position of the joints. The more acute that the angle is, between the plane and the bone, the more likely that the joint has been estimated correctly. Based on this, we estimate that the probability of the joint, that is being empirically identified, due to the bone angle, is calculated as $P(A) = |\cos\theta||\cos\phi|$, where θ is the angle between the bone corresponding to the joint and the xz-plane, and ϕ is the angle between the bone and yz-plane.

Bone Length We know that the person's skeleton is a fixed size, so there should be little to no variation in the person's bone size, from frame to frame; the length of the bone between 2 consecutive skeletons, from the same camera, must be almost identical, in order for it to be a valid joint. Based on this, we can estimate the probability of the joint, based on the bone length variation $P(B) = 1 - \frac{|L_i - L_{i-1}|}{max(L_i, L_{i-1})}\sigma$: where L_i is length of the bone in the current frame, L_{i-1} is the length of the bone in the previous frame, and σ is the scale factor that is set to 5, allowing up to 20% variation.

All of the probability calculations are bound in $[0, 1]$, to ensure that there is no overflow. By combining all of the conditional probabilities of these factors, we can estimate the overall probability of the authenticity of the joint. Since the factors are independent of each other, $PAJ = P(\overline{O})P(L)P(A)P(B)$.

For all of the joints, and for all of the detected skeletons from the cameras, we compute the $P(j)$.

4.3.4 Side Reassignment

The Kinect skeleton uses local features to determine the position of the joints, and as a result, the entire direction of the person is lost. So, the Kinect skeleton tracker assumes that the person is always facing the camera. This assumption is not true in our multiple Kinect camera setup; the person is completely surrounded by cameras, in order to capture

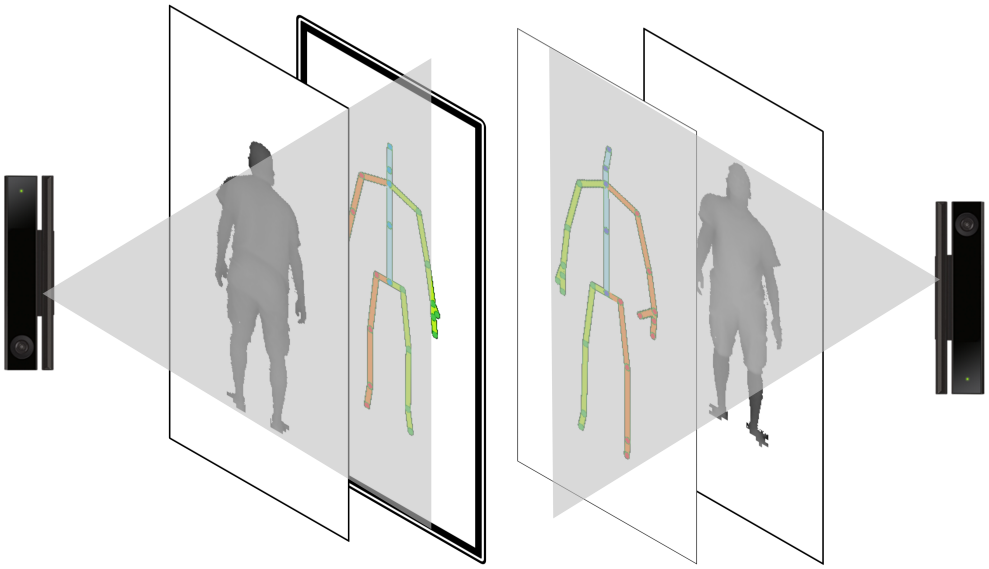

Figure 4.4. The camera on the right tags the right side of the skeleton (red) and the left (green); even after applying a camera extrinsic to change it to a left camera point of view, the joint is still incorrectly tagged, thereby requiring manual side reassignment.

a complete view of them. The Kinects that are set up to capture the person from behind, create problems with how the body joints will be tagged by the Kinects, for the left and right sides. Given that the Kinect assumes the person is always facing the camera, (and these Kinects are actually capturing the person from behind), the Kinect will incorrectly tag all of the body joints exactly the opposite of what they are; all of the joints that are tagged as being on the right side, will need to be changed, or reassigned, to the left, and vice versa.

Side Assignment Procedure: If we are given a set of joints that require the side information and direction of the person, by using the hip center joint for reference, we start tagging the joints to the right of the hip center as the right hip and right shoulder, and the left side as the left hip and left shoulder. After the torso region joints are assigned sides, the side information is propagated to all of the child joints, like the knee, elbow, wrist, ankle, etc. This side assignment strategy ensures that even crossed arms, or legs, are tagged accurately. In the case of a single Kinect, the primary assumption is that the person is facing the camera, even though that may not be true, Figure 4.4 shows how the side assignments are done.

Since the joints are tagged based on their individual proximity with respect to the camera view, the same joint can be considered right or left, depending on the location of the camera. Even after transforming all of the skeletons to the same primary camera point of view, by using extrinsic calibrations, the sides can still be assigned incorrectly, as seen in Figure 4.4. To ensure that the joint sides are marked accurately, the joints are reassigned using the procedure mentioned above, after all of the skeletons are transformed to the primary camera view.

4.3.5 Consensus Skeleton

To determine the skeleton, from the various joint positions from different Kinects, we use a distance constraint $totalPAJ$ maximization approach. The set of joints deemed to be accurate, i.e. $PAJ > 0$, are selected, and then grouped, based on their individual tags. Next, starting from the hip center, each group of joints are combined using the Algorithm 1, with the next group being selected based on the hierarchy of the joints in the skeleton. The joint state, of each joint, is determined based on the γ returned by the algorithm. A threshold of 1 was used, in deeming whether a joint was tracked correctly by JAC. If the root joint, such as the hip center, is declared not tracked by the algorithm, then the skeleton is completely invalidated by JAC: meaning there may not be any person inside the capture area.

The joint consensus algorithm uses a forward selection based strategy, to select the seed point that returns the optimum reward. The reward function is defined as γ, which is just a ratio of the total selected joint positions PAJs, and the rejected PAJs. The approach returns the result immediately, if the γ is greater than the Π. We use a high value of 2 for Π, ensuring early termination, only if there is a two thirds majority in the selected set. The small number of cameras, that actually return good PAJ joints, enable us to retain a high threshold, without compromising on the run time performance of the algorithm. We use a

Algorithm 1 Joint Consensus

```
1: procedure JointConsensus(jointPositions[], PAJs[], startIndex = 0)
2:     center ← jointPositions[startIndex]; n ← jointPositions.size()
3:     tP ← PAJs[startIndex]; mP ← 0
4:     for i ← 1...n do
5:         if i = startIndex then continue
6:         end if
7:         cdist ← distance(jointPositions[i], center)
8:         if cdist < Σ then
9:             center ← (cdist * tP + jointPositions[i])/(tP + PAJs[i])
10:            tP ← tP + PAJs[i]
11:        else
12:            mP ← mP + PAJs[i]
13:        end if
14:    end for
15:    if mP > 0 then
16:        γ ← tP/mP
17:        if γ < Π AND startIndex < n − 1 then
18:            newCenter, γ' ← JointConsensus(jointPositions, PAJs, startIndex + 1)
19:            if γ' > γ then return (newCenter, γ')
20:            end if
21:        end if
22:    else
23:        γ ← tP
24:    end ifreturn (center, γ)
25: end procedure
```

Σ of $5cm$, to ensure that the joints affected by calibration, or fast motion, still contribute to the overall joint estimation. By using only the detected joints, from various cameras, to determine the overall skeleton, it ensures that the consensus approach runs very fast, even for large sets of overlapping cameras.

4.3.6 Person Direction

The direction of the person is very important, in determining the side of most of the joints in the body. Since we use a set of Kinects to capture the user in an open space, we cannot really draw any information about the direction of the person from the capture scene, just

as in the case of the single Kinect. Since the person can twist their body or turn their head, the direction of the person is not always the same as the direction of their face. To determine the person's direction, we follow a cascaded approach.

Direction of Feet: The easiest way to determine the direction of the person is to use the direction of their feet. The direction of the person is the same as, the direction of the vector passing through the point of intersection of the feet, and bisecting the angle between them. If the feet are parallel to each other, then the direction is the same as the feet. Unfortunately, due to ground noise, and its default assumption that the person is facing the Kinect, the consensus approach generally does not accurately identify the feet.

Knee Direction: When visible, the knee joint is almost always at an angle, due to the size variations between the lower and upper parts of the leg. Since the feet have a very little degree of freedom, without the rotation of the knee, it is possible to determine the direction of the feet, just by using the direction of the hip to knee bone if the bones are angular at the knee. We can further locate the feet, by region growing in the same direction from the ankle joint; the centroid of the grown region can serve as a good estimate of the feet.

Curvature around Knee: If there is no noticeable angle at the knee, then it means the person is standing straight. In this situation, if we study the curvature of the surface extracted from the depth image around the knee region, it is possible to estimate knee direction. Due to the limited degree of freedom, muscles around the knee region form a concave side (rear part), and a convex side (front part). To estimate the curvature, we select eight points around the knee, orthogonal to the bone. We further sample another eight points, using a fast detector like pattern (Rosten and Drummond, 2006). To avoid the interference from noise, a wide selector is used to select a single point in a $16x16$ neighborhood; each point is then placed around the original point, to form a $3x3$ monge patch. Finally, the curvature of the center point in the patch is estimated. A Monge patch is a patch $\mathbf{x} : U \rightarrow R^3$, of the form $\mathbf{x}(u, v) = \begin{bmatrix} u & v & f(u, v) \end{bmatrix}^T$; the form where U is an open set in R^2, and $f : U \rightarrow R$

is a differentiable function. Corresponding gaussian curvature is computed as:

$$K = \frac{f_{uu}f_{vv} - f_{uv}^2}{(1 + {f_u}^2 + {f_v}^2)^2} \tag{4.1}$$

where the subscript denotes the partial differentiation $f_u = \frac{\partial f}{\partial u}$. For faster performance, the entire computation can be performed on the GPU. The curvature value at each point indicates a saddle point of 0, whereas it is a concave surface if greater than 0, and convex if less than 0. Using this information, we select two opposing points, one concave and one convex with high magnitudes, and use the convex point as the direction of the leg. If required, a side fill approach can be used at the knee, to determine the size of the feet, if they are not visible.

Head based torso direction: There is a likelihood that the knee joint may not be visible, or that cloth deformations are causing the curvature estimation to yield no good pair. In such scenarios, we use the torso region to determine the person's direction. By using the hip and shoulder joint positions, it is possible to get the axis of the torso region, by fitting a plane through these points, and using the normal information. A faster way is to estimate the normal, is by triangulating the points with respect to the centroid. The normal, at the centroid, is the axis of the torso region. We use the disparity, in the size of the front and rear part of the head, to determine the direction of the person on the axis. As shown in Figure 4.5, even when a person turns their head, the front part of the head always protrudes outward from the neck, or the body.

Once the direction of the person is determined, JAC assigns the appropriate sides to the joints of the skeleton, using the approach described in Section 4.3.4.

4.3.7 Skeleton Accuracy

The Kinect generates a skeleton at $30fps$; even over shorter periods of time, thousands of skeletons are estimated. It is important to verify each and every skeleton, that is generated

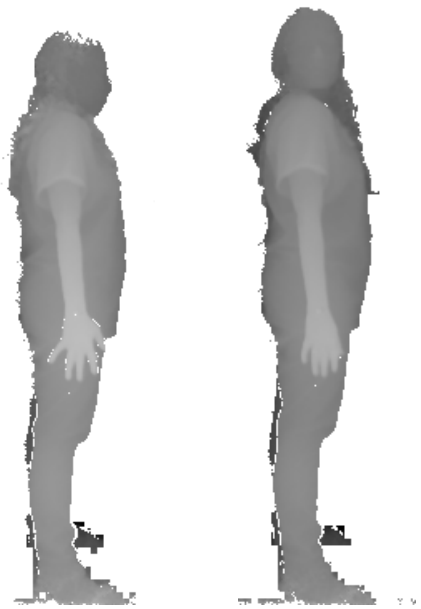

Figure 4.5. Person direction estimation using the head and torso region: The front part of the head protrudes outwards more than the back, irrespective of looking forward (left) or sideways (right).

by the system. Due to the noisy nature of the data, and the bulging nature of muscles in the human body, it is very difficult to generate good ground truth information about the skeleton, using sensor based motion capture systems. Since we are more interested in an accurate description of the pose of the person, rather than the precise positioning of the bones, we define a novel empirical approach to measure the correctness of the skeleton.

For a given 3D skeleton S, of a person captured by cameras $K_1...K_C$, into depth images $D_1...D_C$, let $s_1...s_c$ be the back projected skeletons, using each of cameras extrinsic and intrinsic parameters onto a depth image. Then we define two scores to determine the accuracy of the skeleton.

Containment Score: Intuitively, it is known that the skeleton of a person is contained inside the body. So, no matter what the angle of the camera is to the person, the skeleton cannot be seen by the camera. This basic fact, as shown in Figure 4.6, is used to determine the containment score. While considering the RGB-D camera, from the camera's point

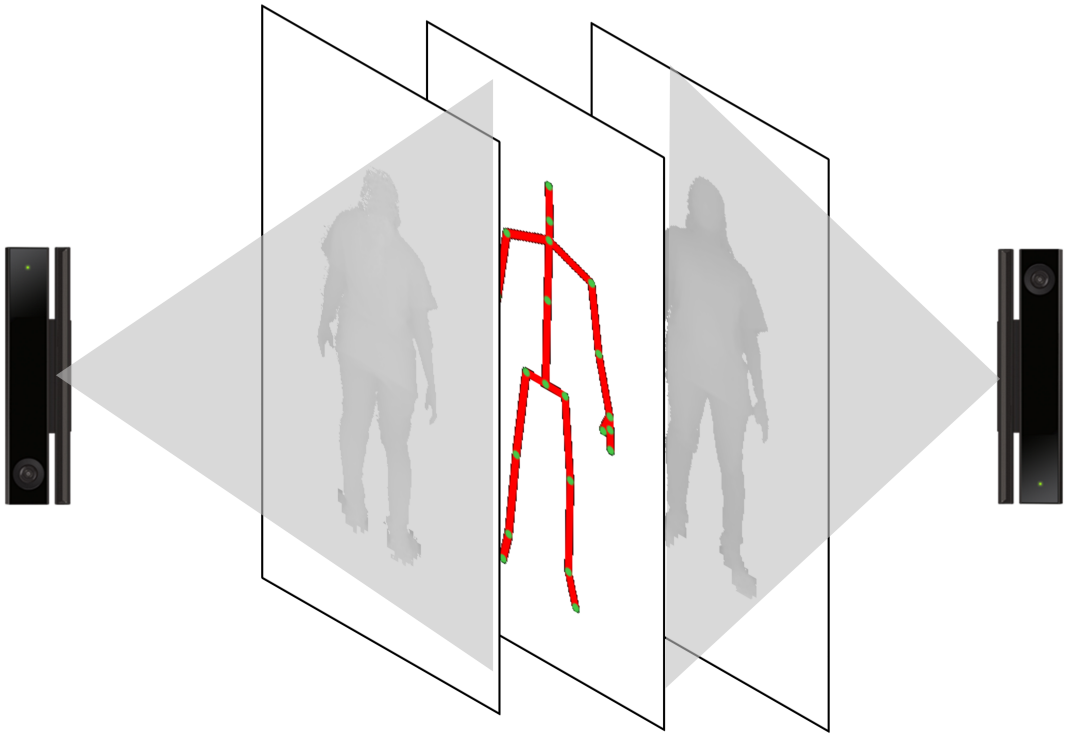

Figure 4.6. The containment score verifies that the skeleton is inside the person, by ensuring that the skeleton is behind the depth data, from all of the camera views.

of view, the depth image should always have a pixel with a smaller depth value than the skeleton, for all of the points on the skeleton. For each camera K_i, a mask image μ_i, with active mask on n_i pixels, is computed using the back projected skeleton s_i. The Kinect score is calculated as

$$\sigma_i = \frac{S_i - \mu_i.D_i}{n_i} \tag{4.2}$$

The final score, $\sigma = \forall imax(\sigma_i)$. The maximum is taken to ensure that, even if one camera view shows the skeleton, then an error is sent to the system. A low containment score means a good skeleton estimate.

Coverage Score: The containment score only measures whether the skeleton is contained inside the captured depth information; it does not verify the size or bone lengths of the skeleton. To ensure that the skeleton covers the complete extent of the body, we define a coverage score. The coverage score measures the percentage of body that each of the skeleton

joints cover. The coverage for each bone is determined by using a Voronoi diagram based partitioning of the point cloud, that is generated using all of the cameras and their depth images. The percentage of points contained in each partition, corresponding to the bone, is then compared with preset ranges for each bone. If a bone coverage percentage is different from the specified range, then the score is calculated as the maximum discrepancy, in the bone coverage percentage of the skeleton.

The coverage score is used primarily as an error indicator, which ensures that manual verification is performed whenever necessary. The coverage score depends heavily on the camera angles, as well as, the overlap between cameras. To avoid incorrect scoring caused by camera overlap points, certain points that might overlap are removed. These points are determined, by projecting each point from the camera K_i, onto the depth image of its neighbors, and checking to see if the pixel value, in the depth image and the estimated depth, are within a $1cm$; if the points are that close to each other, then only one of them is added to the point cloud.

4.3.8 Implementation

Our 3DTI setup uses multiple machines in a large capture area, where the person can perform activities. The person was captured in an open scene ($32x14sqft$ space), using the camera setup shown in Figure 4.7. All of the machines were connected by a single gigabit switch. The bulk of the processing was performed on a single estimation machine, with an Intel i7 2.4GHz processor, 32GB RAM, and GTX 970 graphics. All 12 camera machines had a Intel Xeon 3.0 GHz processor, 18GB RAM, and Nvidia Quadro 4000 graphics. All of the machines were running Windows 8.1 64bit version, and had a Microsoft Kinect V2. The code was written in C++. and unmanaged api of the Microsoft Kinect V2 SDK was used. We use the intrinsic calibration of the Kinect, provided by the Microsoft Kinect SDK. The extrinsic calibration is performed, using two cameras at a time. A planar intersection based

Figure 4.7. The capture area with 12 Kinect cameras, that was used in our experiments.

calibration approach (Auvinet et al., 2012) is used, to get the extrinsic calibration between the two cameras.

In our real-time setup, JAC is implemented in a distributed manner, to achieve faster performance and to reduce transmission times. Calculating $P(\overline{O})$, requires going through the entire person region of the depth image. To optimize this operation, we calculate the $P(\overline{O})$ for all of the skeleton joints for a Kinect, at the same time. This requires between $3ms$ to $6ms$ for the entire skeleton, depending on the posture of the person, captured from the camera angle. The PAJ, for the entire skeleton, takes about $1ms$ more than the $P(\overline{O})$. All of these computations are carried out in the camera machines, and the joint positions along with the PAJ, are transmitted to the estimation machine. Due to the small size of data, it takes less than $1ms$ for the data to arrive at the estimation machine. The consensus skeleton is generated in $1ms$, once all of the Kinect machines' data has arrived (about a $35ms$ window, depending on when the Kinect captures the skeleton). The person's direction, along with

the feet position estimation, takes $1ms$ if only using the knee angle, $4ms$ if using depth based knee curvature, and $5ms$ if using the torso face region. So in the live environment, JAC is capable of generating a skeleton, even in the worst case scenario, in under $12ms$, if all of the cameras capture at the same time. The maximum time taken by JAC to estimate a skeleton, while processing the data set, was $9ms$. The entire data set was stored and processed on the estimation machine.

4.4 Body Sensor Enhanced Skeleton

The multiple Kinect skeleton approach uses the redundancy that is created by the overlapping field of use, of the different Kinects, to overcome the problem of occlusion, as it pertains to pose estimation. In certain scenarios, like when external objects are present inside the scene, it is not possible to have an unobstructed view of the user. In such cases, even the multiple Kinect approach fails to provide a good estimation of the skeleton. Body sensors are devices which can be attached to various parts of the body, to provide specific measurements regarding that region of the body. So, specific sensors, that can measure the position or orientation accurately in the real world, can be attached to the body to improve the accuracy of skeleton identification.

An inertial Measurement Unit (IMU) sensor consists of a tri-axes accelerometer, gyroscope, and campus. Using an IMU sensor, it is possible to determine the orientation of the sensor in the real world, quite accurately. Since the sensor is attached to the body, and is not dependent on some external frame of reference or camera, the measurements are not affected by occlusion. Madgwick et al. use the various readings from the IMU sensor to accurately estimate the sensor orientation, using a gradient descent approach (Madgwick et al., 2011). However, the IMU sensors, as well as the method itself, are highly susceptible to drift, when under heavy motion, or when in rooms with large metal objects.

The drift, in the orientation estimation of the sensor, can be overcome easily by re-initializing the algorithm, whenever drift is detected; this would require the creation of a detection algorithm, and also causes the loss of reading during initialization. Instead of using a detect-and-fix strategy, we use a multiple interval based initialization strategy, to overcome sensor drift related errors. Multiple instances of the (Madgwick et al., 2011) approach are initialized using the sensor data at fixed intervals, each running for an average duration of about 50 seconds (around $10,000$ samples); it was noticed that typically, the orientation estimation approach drifted away after about 83 seconds. A new instance of the orientation approach was started every $5,000$ samples, allowing the newer instance to take over the previous instance, once it had reached the $10,000$ sample threshold.

Camera based approaches can provide a good estimation of the overall skeleton pose, but are terrible at estimating the orientation of a bone, specifically the rotation along the bone axis. In conditions requiring the accurate tracking of the joint orientation in all directions, it is always better to use an IMU sensor. A hybrid skeleton can be created by combining the camera based skeleton, with the orientation information, that is provided by the body sensors that are attached to the body. The bone length, for the bones with the IMU sensor on them, are obtained by averaging the bone lengths returned by the Kinect, but only when both of the orientations, of the Kinect skeleton and the IMU sensor, match. After the bone lengths are identified, the skeleton is always modified, based on the orientation of the IMU sensor, whenever it is available; for the rest of the bones, the camera based skeleton estimation is used. Figure 4.8 shows the Kinect skeleton, and the accurate hybrid skeleton that is created by using the IMU sensors on the arm.

4.5 Haptic Enhanced Skeleton

The i3DTI framework is easy to extend, and is capable of supporting many different kinds of devices. The haptic device is used to provide force feedback to the user, while interacting

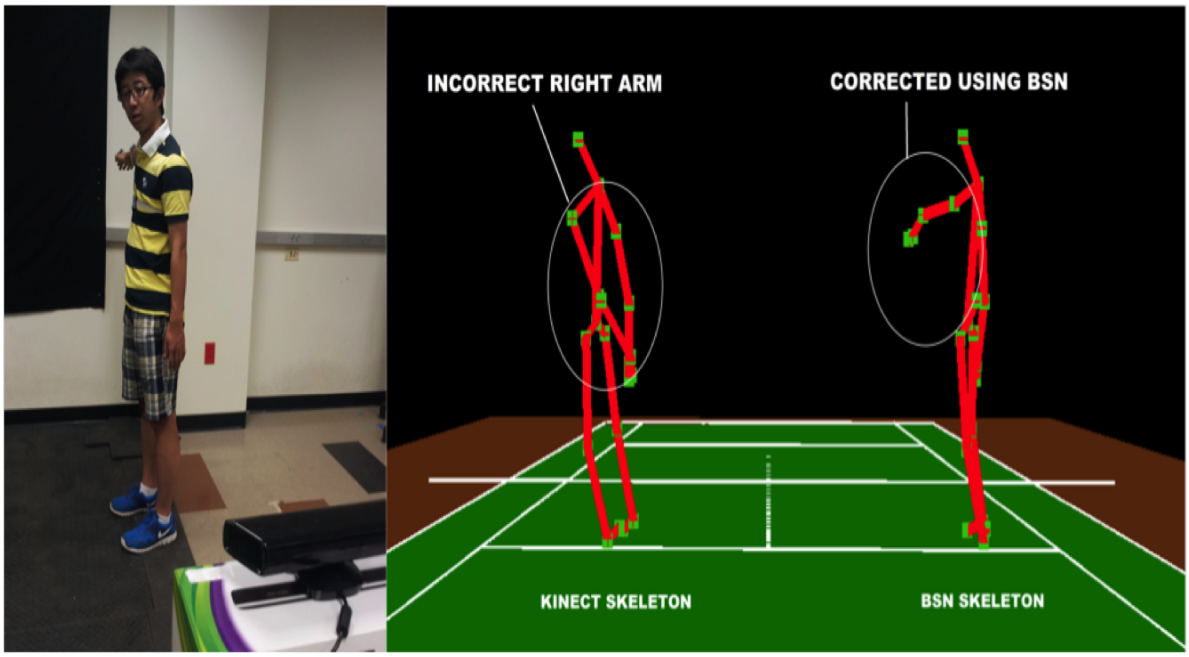

Figure 4.8. Combining the Kinect skeleton with the bone information from the IMU sensor.

with the system. For the force feedback to function properly, the haptic device must be placed close to the user. In most situations, the close proximity of the haptic device to the user, causes it to occlude at least the person's hand. Even the multi-connect skeleton estimation approach can not accurately determine the position, or orientation, of the user's hand, while it is occluded by the haptic device. It is possible to estimate the orientation of the user's hand using the body sensor approach, but given the large amount of occlusion, multiple body sensors would be required to get a reasonable position estimate; for this reason, it is more appropriate to estimate the user's hand position just by using the haptic device.

When using the haptic device, it is assumed that the hand will usually be occluded by the device; since the hand holds onto the haptic handle, the position of the handle can be used to estimate the position of the hand. The haptic device has its own coordinate system, and the cameras have their own $R3$ space. In order to calibrate the two spaces, common points need to be extracted in both, the camera and haptic, spaces. When multiple cameras are used to capture the user, a color marker placed on the haptic handle, as shown in Figure 4.9, can be

Figure 4.9. Hand joint estimation using the position of the haptic.

detected and tracked to extract points for the camera space. In scenarios with only a front facing camera, with no clear view of the haptic handle, a temporary camera, having a clear view of the handle, is introduced into the scene. These two cameras are then calibrated, using the approach described in Chapter 2. The points extracted from the camera space, together with the corresponding haptic positions, are then registered using Horn's method (Horn et al., 1988).

After registering the camera and haptic space, the transformation matrix is then used to calculate the position of the haptic handle, in the camera space. So whenever the haptic device is in use, the position of the user's hand is interpreted to be the same as the position of the haptic handle, in the camera space. The haptic based hand position estimation, when combined with the camera based skeleton, provides an accurate representation of the user's pose. This hybrid skeleton is used with various interaction techniques, to provide highly responsive, natural user interactions, for i3DTI systems.

PART II

COMMUNICATION

An i3DTI application needs to transmit multi-modal data over the internet. 3D mesh data is reconstructed every frame and contributes the majority of data being transmitted. This data is typically transmitted either as a set of depth and color images, or a mesh and texture image. The color/texture image is compressed using JPEG, while the rest of the data is compressed using zlib (Salomon, 2007), which significantly reduces the data size. But even with the compression, the data size per frame is still substantial, causing an increase in latency and a drop in frame rate. In the communication part, we present approaches that can transmit sequences of live captured 3D user mesh data, with minimal latency and maximum frame rate over the internet.

Chapter 5: Given the large size of the mesh data, it is necessary to reduce the amount of data that needs to be transmitted every frame. The skeleton prediction approach relies on the user's skeleton to predict the state of the user's mesh at the sender's side. A previously received mesh is deformed using the latest skeleton, and then rendered on the receiving side.

Chapter 6: The skeleton prediction approach does not have any selection mechanism for the mesh used for deformation. The choice of the mesh to be deformed is based purely on network conditions, and leads to artifacts in the predicted mesh. The DIstortion Score POse SElection (DISPOSE) approach eliminates these artifacts, by providing a selection strategy for each of the meshes to be transmitted and used for prediction for each user pose.

Chapter 7: Since there is no restriction on the mesh that is deformed based on the skeleton, it is possible for anybody to capture and manipulate other peoples' meshes to create realistic videos. The security impact of such manipulations, and the approaches to detect them, are discussed in this chapter.

CHAPTER 5

SKELETON PREDICTION

3D tele-immersion(3DTI) allows users from distant sites to collaborate in a single virtual space, using their real life 3D avatar. It also enables interaction with shared 3D objects in the virtual space, in real time. 3DTI has potential applications in remote training (Wu et al., 2008), rehabilitation (Keshner and Kenyon, 2004), entertainment (Yang et al., 2006), games (Raghuraman et al., 2012), etc. It combines many aspects of computer vision for image acquisition, graphics for reconstruction and rendering, virtual reality for display, and networking for high data transmission.

There are many challenges associated with providing such a system. A single frame of a person, captured by a single camera, has around 100,000 points of information; this results in a frame size of approximately 1.5 MBytes. A single site uses many such cameras to attain high visual quality. To send such massive volumes of data across the internet, while maintaining the quality of experience (QoE), is one of the major challenges in 3DTI systems. QoE consists of both the visual quality as well as the rendering rate, which is measured in frames per second (fps). These QoE requirements are often contradictory, since providing higher visual quality often results in lower fps and vice-versa; so, sending a detailed mesh, with a huge volume of data over the internet, results in excellent visual quality and accuracy, but also leads to low frame rate, causing poor overall QoE.

Current methods used to deal with this problem involve compressing the data in some way. Some methods use standard compression libraries directly on the color and depth data (Redert et al., 2002). Some of the methods use a combination of standard compression libraries to compress the data before it is sent over the internet (Yang et al., 2006). (Shi et al., 2009) introduces a view-dependent real-time compression scheme geared towards use in mobile devices. (Kum and Mayer-Patel, 2005) uses a reference stream to achieve inter-stream compression. (Lien et al., 2009) uses a model based approach, where significant

points (skeletal joints) are extracted from the point cloud frame, and then the compression is achieved through motion estimation. All of these compression schemes are able to achieve approx 10-15 fps over the internet.

Our approach uses the inherent redundancy in the data being transmitted, by predicting and animating the mesh at the receiver side based on a few points of information, rather than always compressing/decompressing data at both the sender/receiver end. In the process, we transmit only a few bytes of data every frame, enabling us to achieve good frame rates. The approach is motivated by three fundamental observations: (1) Transmitting large amounts of transient information over the internet is not feasible; however, it is possible to stream small pieces, less than 1KB, of information at a higher rate, even on a slow internet connection. (2) 3D deformations are estimated by the impact of force at a point, resulting in the motion of the points around it. By streaming only the locations of these key points and their relationships with their neighbors, the entire scene can be reconstructed at the other end (Tang et al., 2010). (3) Also, the skeleton of any object, in its most deformed state, can be used to identify the points that influence the entire object (Yoshizawa et al., 2003). For example, in the case of a human, the joints of the body signify all of the aspects of the bodily deformation. By combining all of these aspects in this paper, a prediction based scheme is introduced that uses already transmitted meshes, as well as skeletal points, to manipulate the mesh at the receiver side. A small transmission of a small number of points can be used to effectively predict the position of millions of points at the receiver side, giving a frame rate equal to the rate at which the data is captured. Details of this method are discussed more in Section 5.1

Our contributions form a novel method for solving the excessive data streaming problem, using a prediction that is based on the information that has already been streamed, by deforming a live model. A fast segmentation and parallel prediction method is introduced to allow the real time prediction of rigid body deformation, based on skeletal points.

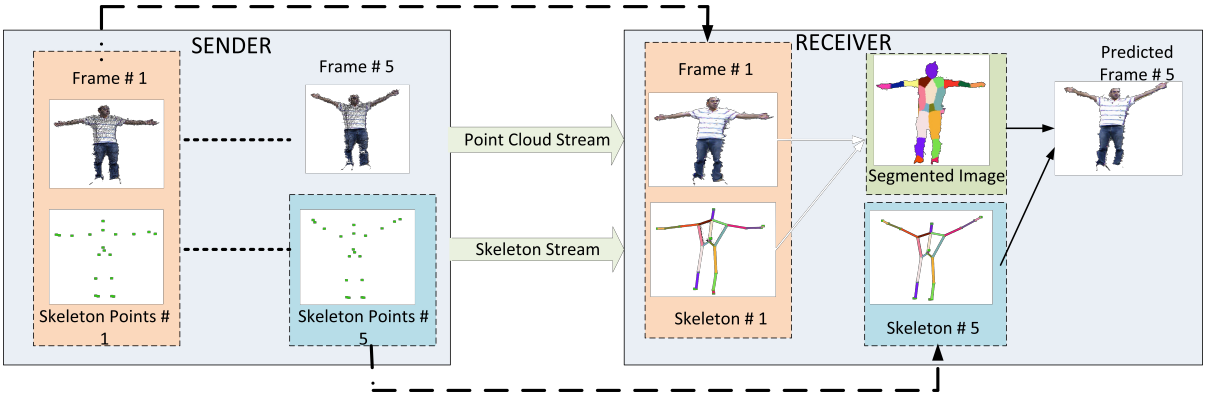

Figure 5.1. Shows the overall sequence of processing for the approach.

5.1 Our Approach

Reducing the amount of data being transferred by the 3DTI system would increase the frame rate and improve the user experience. To achieve this goal, our approach uses a multiple stage process, as shown in Figure 5.1. In the first stage, the skeletal points are identified from the depth image at the transmitter side. These points, along with the mesh, are transmitted to the receiver the first time. After this, the meshes are only sent if the network bandwidth is sufficient to send it; subsequently, only the skeletal points are transmitted.

At the receiver side, the mesh is segmented based on the skeleton that was generated using the skeletal points. This segmentation is stored and used to predict the behavior of all the other points inside the region, based on the movement of the skeletal points. Finally, the transformation of the segmented region is estimated and applied in accordance with the skeletal point changes. Each of these steps are described in greater detail in the following sections.

5.1.1 Skeleton Identification

3DTI usually involves transmitting the user's mesh, so in this paper we consider only human skeletons; so we use a human skeleton identification approach, described in (Shotton et al., 2011). A computer vision based system is used to identify the limbs of the person. Features

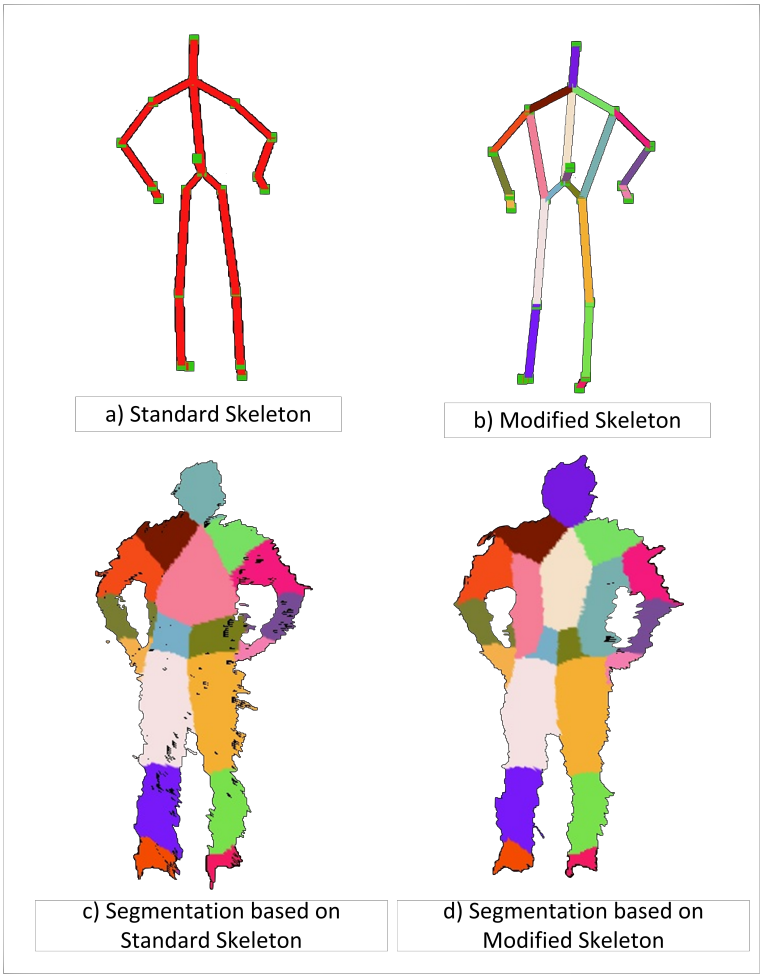

Figure 5.2. Skeletons and their corresponding mesh segmentation

are extracted from the depth image, and a random decision forest algorithm is used to identify the various parts of body from these features. Once the broad level limbs have been identified, a skeleton that is based on these limbs is fit to the overall model. The skeleton, shown in Figure 5.2(a), is used. The estimated joint positions from this skeleton fitting are used in our approach.

The standard skeleton, as shown in Figure 5.2(a), is sufficient for most representations of the human body, but it cannot track the rotation of the skeletal segment about its axis. A few extra skeletal segments are added to allow for such motions to be tracked. Since only the point positions are transmitted, the extra information is generated at the receiver side

from these points, allowing for small packets. The modified skeleton uses more segments to represent the mesh of a human, as shown in Figure 5.2(b).

5.1.2 Segmentation

The entire mesh is segmented into different Segmented Regions(SR), based on the distance between the vertex and the skeletal segment. Each skeletal segment has a control point $P_c(x_c, y_c, z_c)$, typically consisting of a skeleton joint and a reference point $P_r(x_r, y_r, z_r)$, which could be any point of the skeleton segment connecting two skeletal joints, or any point inside the segmented region. To reduce the transmitted data and enable accurate tracking of the SR, the control point of the next SR is used as the reference point whenever possible.

A Voronoi decomposition (Aurenhammer, 1991) based approach is used to segment the regions around the line segment (P_c, P_r), by estimating the distance between a point and the line. For a given point $P(x, y, z)$, and a line segment (P_c, P_r), the distance d is given by

$$\begin{aligned} d &= \begin{cases} ||P - P_r|| & \text{if } t \geq 1 \\ ||P_c - P|| & \text{if } t \leq 0 \\ \frac{||(P-P_c)x(P-P_r)||}{||P_c-P_r||} & \text{if } 0 < t < 1 \end{cases} \\ t &= \frac{(P_c - P).(P_c - P_r)}{||P_c - P_r||^2} \end{aligned} \tag{5.1}$$

where t signifies the projection of the point on the line segment. Using the only the distance metric, the entire mesh is segmented, as shown in Figure5.2(d). Using the skeleton with extra segments allows for better segmentation of the mesh, since the segmentation model is based on skeletal segments. It also allows us to capture the deformation that is occurring due to the rotation of points around the line as the axis, which normally cannot be captured by the line segment based representation.

5.1.3 Prediction

The segmented regions are treated as individual rigid bodies that can undergo transformations independent of each other. A local spherical space, with the control point at its origin is used to represent each point in the SR. A point in $P(x, y, z)$ is represented in the local spherical space as $P_s(r, \theta, \phi)$, where

$$\begin{aligned} r &= |P_c - P| \\ \theta &= \arccos \frac{z - z_c}{r} \\ \phi &= \arctan \frac{y - y_c}{x - x_c} \end{aligned} \tag{5.2}$$

Every SR for the received meshes is represented this way. The spherical value for the reference point of a SR is represented as $P_rs(r_{rs}, \theta_{rs}, \phi_{rs})$; for the new value of control point P_{cn} and reference point P_{rn}, a new spherical value P_{rns} is estimated for P_{rn} using P_{cn} as the center. Since we assume a rigid body, the value of r is not going to change for any of the points. The new spherical value P_{sn}, for a point P_s, is given by $(r, \theta + (\theta_{rns} - \theta_{rs}), \phi + (\phi_{rns} - \phi_{rs}))$, once all of the points in all of the SRs are updated. The new point in 3D space $P_n(x_n, y_n, z_n)$ is given by

$$\begin{aligned} x_n &= r_n \sin\theta_n \cos\phi_n + x_{cn} \\ y_n &= r_n \sin\theta_n \sin\phi_n + y_{cn} \\ z_n &= r_n \cos\theta_n + z_{cn} \end{aligned} \tag{5.3}$$

The prediction method relies on the relationship between the point and the control point of the segment. The transformation is applied based on the changes to the skeletal segment, on which the reference point is situated. A direct mapping relationship allows for fast paralleled computations. This allows the prediction to run extremely fast on a large number of points.

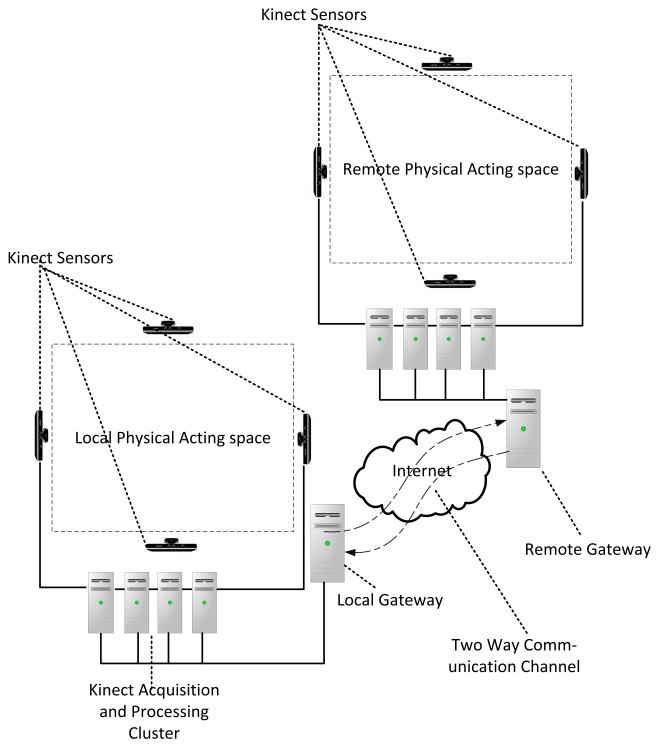

Figure 5.3. Architecture of the 3D Tele-immersive system

5.2 Implementation

The architecture of the 3DTI system is shown in Figure 5.3. Each site consists of a renderer, a gateway, and a set of camera machines. Multiple cameras are used to capture the person at each site, and 3D reconstruction is performed on the capturing machine. One or more cameras are connected to a camera machine, based on the data bandwidth available. In our experiments, a single Kinect is used on the camera machine. The vision based skeleton detection approach generates a confidence score for each identified skeletal point. The skeleton from the camera, with the highest cumulative score, is used as the skeleton for that person on the site.

All of the camera machines transmit the information to a central gateway. This acts as a channel between the different sites and maintains the synchronization and flow of data. A modified version of the stop and wait protocol is used to transmit information across

the network to the remote site. Every time a mesh frame is sent out, the system waits for an acknowledgment (ACK); once the ACK is received, the most recent packet is sent out. Deflate (Salomon, 2007) is used to compress the mesh information and, on average, reduces the size of the data by 50-75 times. The skeleton points are transmitted as soon as they are generated.

The rendering machine combines the information coming in from various local sources, and from remote sources through the gateway, to generate the meshes to be rendered. It also performs the segmentation based on the skeleton and mesh information it receives. For proper segmentation, the renderer maintains a skeletal buffer, in order to match the mesh with the skeleton it received in the past. Every time a skeleton frame is received, the renderer applies the prediction algorithm on the points associated with that skeleton, based on the initial segmentation.

5.3 Experiments

A variety of experiments were performed to test the results of each stage, and also to gauge the effectiveness of the approach in a real life situation. All of the experiments used a single Kinect camera for the acquisition of the mesh data at a site. In order to allow for better visual comparison, the rendering view was kept similar to the view of the capturing cameras for all comparisons. Standard qualitative measures cannot be directly applied here, since two consecutive frames of a static scene, using a Kinect camera, show significant differences; hence, it is not possible for direct one-to-one point level mapping, between the frames, without approximations. Consequently, it is also not possible to compare the predicted result for each point. So, the metric of Quality of Experience (QoE), in this case visual experience, was used to measure the performance of the approach, as described in (Wu et al., 2011). All of the participants dressed normally, and there was no special care taken to avoid loose clothing.

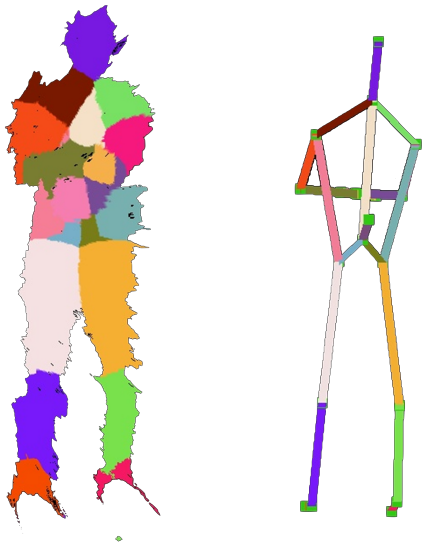

Figure 5.4. Incorrect mesh segmentation and its corresponding skeleton for folded hands

To validate the segmentation part of the approach, poses extracted from six different activities were used: waving, punching, kicking, jumping jack, hands in pocket, and folded hands. A visual validation scheme was used to gauge the correctness of the segmentation. All mesh segments were colored as shown in Figure 5.2(d), and visually verified. It was noticed that the standard skeleton, shown in Figure 5.2(c), performed poorly in almost all of the poses, except when the hands were away from the body. The modified skeleton, shown in Figure 5.2(d), performed well in all of the cases, except when the hands were very close to the body, especially when the hands were folded, as shown in Figure 5.4.

Meshes generated at various frame rates (1 to 30fps) were used to validate the effectiveness of the prediction method. The same six actions were used. Numerous gaps were noticed when using the point cloud representation for rendering the data, but these gaps soon disappeared with the use of a mesh. A sample activity of hand waving, at various frame rates, is shown in Figure 5.5. There is a direct correlation between the initial pose that is used to segment the points, and the future predicted mesh based on the activities performed. If the segmentation is perfect, then the predicted meshes are also very similar; but if either the skeleton determination or the segmentation perform badly, then the prediction results are

Figure 5.5. Prediction sequence based on the first frame for a one second period at 5 fps. Top row shows the actual data and bottom row the predicted result.

very poor too. Therefore, it is important to select the right key poses that the predictions should be based on. In this paper, no such determination is made, and all of the predictions are based on the mesh information that is made available.

Finally, a virtual training setup was used to test the real world effectiveness of the solution. Both the sites used standard cable internet, with a 10mbps down and 512kbps up connection. The trainer asked the users to perform the six activities listed above. The trainer at the other end then looked at the results side by side in real time, one with the prediction and the other without, having to decide which looked better. It was observed that the users preferred the prediction based methods in most of the cases, except the activity involving the folded hands, and a few with their hands in their pockets. It is evident that the prediction based scheme performs better in real world conditions, except when occlusion causes incorrect skeleton detection and segmentation.

5.3.1 Comparison With Existing Approaches

Existing approaches use compression to solve the problem of reducing the amount of data to be transferred to increase the frame rate. Since our approach uses a prediction based method

to solve the same problem, a direct comparison cannot be drawn between the approaches. However, we can compare the number of bytes transferred per frame using each method, over a period of time per camera. At a camera level, all of the systems generate an equal amount of information per frame. For compression based techniques, all of the generated frames need to be transmitted, thereby reducing the frame rate; Whereas for the skeleton prediction technique, streaming only a few mesh frames might yield high frame rates that result in better quality rendering. The data is reduced in size by approximately 5:1 in (Yang et al., 2006), 15:1 in (Kum and Mayer-Patel, 2005), and 50:1 in (Lien et al., 2009), with residuals results in about 1-2, 3-6, and 10-15 fps, respectively. It is possible to achieve 30 fps(rate of capture) using the non residual version of (Lien et al., 2009), by degrading the rendering quality dramatically. However, in using our method, we achieved that same frame rate without significant degradation.

CHAPTER 6

DISTORTION SCORE BASED POSE SELECTION

3D Tele-immersion (3DTI) systems are used in a wide range of applications like telepresence, tele-medicine, physical rehabilitation, gaming, teleart, etc. (Kurillo and Bajcsy, 2013). 3DTI systems allow real time collaboration between people at different locations, by rendering their "live" avatars in an interactive virtual world; this is achieved by using multiple cameras to capture the scene at each location. Every frame of captured information is processed to generate 3D models, which are then transmitted over the internet, and rendered in the virtual world. The generated meshes are large in size (about 6 MB per camera), consisting of thousands of vertices and triangles. Even on high speed networks, after lossy/lossless compression, the transmission frame rate averages around 10 fps, resulting in a jittery rendering at the other side.

Skeleton based 3DTI transmission schemes (Lien et al., 2007; Raghuraman et al., 2013) overcome the issues of low frame rates that are present in the compression based systems, by using skeletal data to animate the sender's 3D mesh. Smaller size skeleton data is transmitted every frame, to allow the receiver to animate the current pose at the sender's side. The large sized 3D meshes are transmitted only intermittently, depending on the network bandwidth. The reason for this is that, any recently transmitted mesh will produce good quality animation for any given skeleton.

However, the situations where the most recently transmitted mesh and skeleton have a high amount of inconsistency, result in poor animation. For example, if the most recently transmitted mesh has a person with folded arms, and the most recent skeleton is in a T-pose, then the resulting animation would be comprised due to the occlusion in the mesh. Therefore, choosing meshes to be transmitted, based only on the availability of network bandwidth, compromises the quality of the animation.

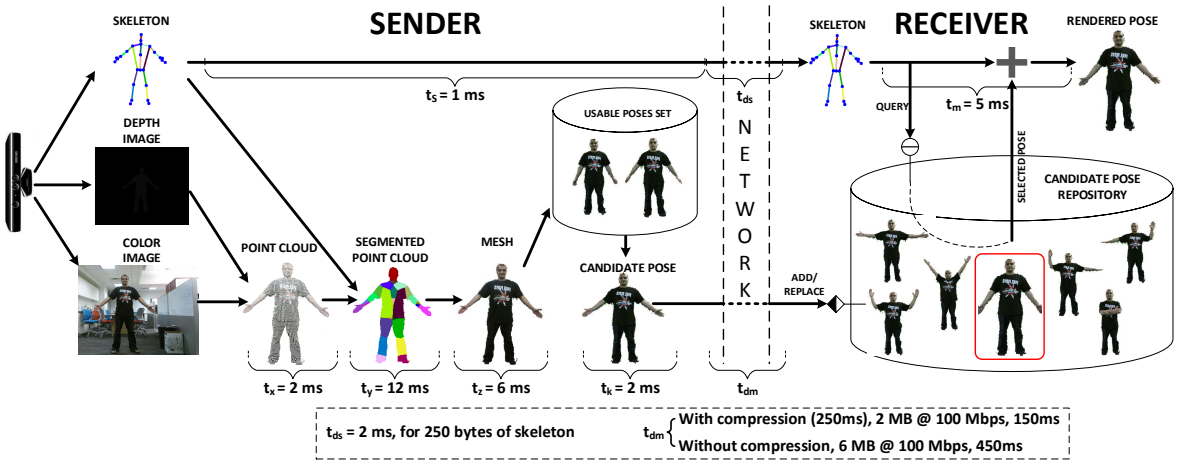

Figure 6.1. The architecture of DISPOSE based 3DPTI.

Pose space, or example based, animation (Lewis et al., 2000; Xu et al., 2011) can successfully generate good quality animation, using only a few key mesh poses. By retaining many more meshes, instead of just using the last received mesh, enables the system to generate more realistic animation. 3DTI systems generate approximately 30 meshes a second. Online pose set generation, in real time, using traditional approaches is not feasible for such large volumes of data.

Proposed Approach: In this paper, we present a DIstortion Score based POse SElection (DISPOSE) approach to generate high quality animation, by selecting the best possible outcome for a given pose. The images captured by the RGB-D cameras, go through a series of processing steps to generate a deformable mesh. Artifacts are caused by various real world factors, like self occlusion, clothing, lighting, etc. Moreover, the captured data that is noisy, along with inaccuracies in the processing pipeline, result in artifacts. DISPOSE models all of these possible artifacts as distortion in the animated mesh. By studying these distortions, DISPOSE decides on the mesh that is most likely to generate the best possible estimation of the pose.

DISPOSE is applied both on the sending and the receiving side of a regular 3DTI setup, as shown in Figure 6.1. At the sender's side, the DISPOSE based 3DTI approach determines

the candidate meshes that need to be transmitted over the network, based on the effectiveness of the mesh to animate different poses. At the receiver's side, the DISPOSE approach also buffers distinct meshes, to allow the animation from the best candidate pose, for that given skeleton. While creating an animation sequence, higher priority is given to the most recently selected pose, in order to retain temporal coherence between the frames, thereby improving the perceived visual quality. The average mesh captured by a single RGB-D camera is about 6MB, requiring 450ms to transmit on a 100 Mbps network. With compression (Yang et al., 2006) needing 250ms of processing time, the mesh size is reduced to about 2MB, which requires 150ms to transmit over a 100 Mbps network. In comparison, for every frame, DISPOSE requires 2ms to transmit the skeleton, and 5ms to animate the mesh at the receiver side.

Our Contributions:

- The real-time quantification of distortions that are introduced during the animation of a captured human 3D model, using RGB-D cameras. The proposed Distortion Score shows linearity with respect to the visual quality of the 3D human meshes.

- Online creation of distinct example poses, suitable for animation in real time.

- The proposed Distortion Score is shown to be effective in 3D tele-immersion both (i) at the sender side for selecting candidate meshes to be transmitted, and (ii) at the receiver side for the selection from a repository of received meshes, so that the animated sequence shows better visual quality.

- Incorporation of the proposed DISPOSE method implies that the latency in rendering the sender side activities, are reduced to the transmission delay that is associated with the skeletal data transmission. Since the skeletal data is about 250 bytes, this latency is indeed very low.

6.1 Related Works

Many methods have been proposed to transmit 3DTI information. Silhoute based meshes are directly transmitted by (Petit et al., 2009). Image and *zlib* compression are used by (Redert et al., 2002) and (Yang et al., 2006) respectively. View-dependent compression was proposed by (Shi et al., 2009), for mobile devices. A heuristic based lossless mesh compression was proposed by (Mekuria et al., 2014). A block difference based stereo camera data compression was proposed by (Zhou et al., 2011). Multi-stream adaptive compression based transmission was proposed by (Yang et al., 2006, 2010). A single merged mesh is transmitted by (Alexiadis et al., 2014; Mekuria et al., 2013; Alexiadis et al., 2013), for improved performance. Multiple views obtained using the Kinect and bisection algorithm (Vasudevan et al., 2011) for meshing, allowed (Kurillo and Bajcsy, 2013) to develop a low latency compression 3DTI approach.

Skeleton based approaches (Lien et al., 2007; Raghuraman et al., 2013) extract the skeleton from the data, and use interpolation to generate the next frame by using the previously arrived information. While skeleton based approaches offer higher frame rates, the visual quality of the result depends on the actions that are being performed, and the transmitted mesh.

Our approach uses multiple meshes to generate the animation at the receiver side. Animation that uses multiple pose meshes (Lewis et al., 2000; Weber et al., 2007) produce good quality results. These methods require set poses and a highly accurate registration in situations involving captured meshes. Mesh to mesh registration takes a significant amount of time (Weber et al., 2007). Alternatively, using a single weighted mesh to generate animations with little artifact, reduces some post processing time(Vaillant et al., 2013). Generating accurate weights for meshes is too time consuming to be used in 3DTI systems. Combination methods, using pose selection and interpolation, are proven to be effective for video generation (Xu et al., 2011); however, these methods require offline mesh pose database creation, and take seconds to synthesize a single frame.

Given that the overall process of generating weights or pose database is time consuming, our method instead focuses on determining the quality of the animation result, to select the visually "optimal" result. Various mesh evaluation metrics have been proposed in literature. A surface approximation of Hausdroff distance was used by (Aspert et al., 2002), to perform more efficiently. A surface roughness based measure was proposed by (Corsini et al., 2007), for comparing watermarked meshes. Metrics based on simple distortion measures, such as Hausdroff distance and root mean square error, do not correlate with the human visual perception. A distortion metric, based on the difference of the structure (captured via curvature statistics) of the meshes being compared, was proposed by (Lavoué, 2011). Local mesh roughness, derived from Gaussian curvature, was proposed by (Wang et al., 2012). Even though these methods correlate well with human visual perception, their computation takes a few seconds for a single mesh.

6.2 3D Tele-Immersion Overview

3D Tele-Immersion systems allow geographically distributed users to be immersed in a unified virtual environment. To display the "live" avatar of the person, multiple cameras are used to capture the user. All of the cameras are calibrated both intrinsically and extrinsically. For each session, depending on the transmission scheme, the following steps are performed each frame:

Acquisition: The user is captured using an array of stereo or RGB-D cameras that are surrounding them. The user is then extracted from the image, using background subtraction.

Meshing: Captured range images are projected to 3D using the intrinsic calibration of the cameras. The local neighborhood information from the range images is used to triangulate points, in creating a single mesh for each camera (Pajarola et al., 2003). These meshes are realigned based on the extrinsic calibration between the cameras. The aligned meshes

are then zippered together using (Turk and Levoy, 1994), to create a single mesh for each user.

Compression and transmission: The generated mesh is compressed using zlib. The texture image is encoded as a jpeg image, similar to (Redert et al., 2002) and (Yang et al., 2006). The data is then transmitted to the receiver for rendering.

Skeleton based 3DTI transmit compressed meshes when the network bandwidth is available. A reference skeleton is transmitted every frame, allowing the receiver to animate the last received mesh into the sender's pose. These are extra operations that are involved:

Skeleton Extraction: The skeleton is provided by the Kinect sensor, using a vision based approach in real time. The extracted skeleton is inaccurate in cases of occlusion. Occlusion related inaccuracies can largely be avoided by combining multiple Kinect skeletons to a single more accurate skeleton (Yeung et al., 2013). Even with multiple Kinects, the skeleton can be inaccurate in situations with self occlusion.

Segmentation: If an image based transmission scheme is used, then the depth images are directly transmitted to the receiver, and the depth image is segmented into regions based on the skeleton. A region growing based approach, as described in (Adams and Bischof, 1994), is used to identify each part of the body. If a mesh is transmitted directly, then a voronoi based approach is used to segment the mesh. The skeleton is projected onto the 3D space. The distance between the skeleton joint and the vertex is used to determine the segmentation.

Skinning: When a skeleton is received, the most recently segmented mesh is used to animate the mesh. Spherical blend skinning, that uses a constant weight for each segment, is used to animate the mesh. Since estimating accurate weights for each vertex is time consuming, a rigid association is assumed for the entire segment.

Figure 6.2. From left to right: The various artifacts *(highlighted)* that are generated by occlusion, rigging, segmentation, meshing and skinning, respectively.

6.3 Visual Quality Challenges

The visual quality of the result is affected by the network bandwidth restrictions, camera calibration, noise from the cameras, depth estimation, image distortion, etc. Typically the data captured from a Kinect is extremely noisy. While processing this noisy data, a lot of new errors are introduced, depending on the type of processing. The noise generated at the source level by the RGB-D cameras, as well as calibration errors between the cameras, are not considered in this paper. We primarily focus on the errors generated by the following:

Occlusion: RGB-D cameras follow the pinhole capture model, which results in a lot of missing elements while capturing a scene. For an item to be captured by the camera, it needs to be in direct view of the camera. Since many of the poses, such as folded hands, cause occlusion, these poses create an empty space in the mesh, as shown in Figure 6.2.

Clothing: Skeletal animation is not used to deform clothes. Loose clothes are simulated, independent of the overall body of the rigged model. The deformation of clothing relies on not just a single frame, but also on a series of events before the current frame. In skeleton based 3DTI systems, the vertices of the clothes are treated the same as the body, which creates large artifacts. After animation, depending on the texture of the clothes, inconsistencies between individual segments of the mesh are also clearly visible.

Rigging: In character animation, rigging should fit the hierarchical bone structure, also called the skeleton, to the mesh accurately. This is typically done by a person, or by a semi-automatic script. The skeleton identification process identifies and fits the skeleton to the depth information, with reasonably good accuracy. However, for many poses, such as the one shown in Figure 6.2, this rigging process can yield bad results, causing a cascading set of errors from various other parts of the system. Rigging error almost always leads to segmentation errors, causing similar artifacts.

Segmentation: It is not always possible to determine the exact boundaries of each limb with a high degree of accuracy. If the segmentation is incorrect (refer Figure 6.2), the animation yielded will also be incorrect, resulting in cobweb artifacts.

Meshing: When the mesh is created from the depth image, it is not possible to determine and segment the exact boundaries of the various objects inside the scene. The segmentation information is not used while creating triangles for the mesh; this can cause the generated triangles to span across limbs, or connect the person to the object near them, as seen in Figure 6.2. When such a mesh is used for animation, cobweb artifacts are generated.

Skinning: Skinning moves the vertices of the mesh to achieve its animation, and is based on certain assumptions. The primary assumption is that all of the deformation, occurring in the mesh, should be a direct result of the changes in the skeleton. The actual influence of the skeleton on the position of the vertex, is interpreted differently in every method, thereby resulting in different artifacts. Figure 6.2 shows the artifact generated, when using the most widely used Linear Blend Skinning (LBS) method. When using LBS, typically the artifacts show a shrinking of the mesh around the joints. For the method used in (Raghuraman et al., 2013), there is a problem with the stretching of the bone joints.

In the later sections, we show that all of the artifacts that are generated can be transformed to a measurable form of meshing artifacts. We also define the Distortion Score, which measures the exact amount of artifact in the mesh. This score can be used to understand the visual quality of the mesh to be rendered.

6.4 DISPOSE

3D tele-immmersion systems generate a large number of meshes every second, only a few of which are useful in animating a mesh by using the skeleton of the participant. In skeletal animation, a rigged mesh can generate a few fixed animations, and the vertex skinning weights of the mesh are tuned to ensure that there are no artifacts during the animation. Despite tuning the associated weights, the animation that is produced by the rigged meshes is accurate only within the bounds of the motion currently being performed. Over time, as new (never seen before) movements are introduced, it is highly likely that the mesh will deform in an undesirable manner, to animate these new actions. Therefore, the quality of animation is directly influenced by the vertex weights and chosen mesh. DIstortion Score based POse SElection (DISPOSE) provides a scheme of selecting the mesh that is the most likely to produce a better quality animation, for the given skeleton. Instead of using a single mesh of a person in a particular pose to animate all other poses, DISPOSE selects a mesh, from a set of meshes, that is most likely to animate the current pose accurately, without any artifacts. The visual quality of the rendered result is enhanced significantly by selecting the most suitable mesh to animate the user's current pose.

We briefly discuss mesh optimality, with respect to estimating poses. We refer to an ideal rigged mesh, as one by which all other meshes can be animated accurately. A rigged mesh, with its weights estimated for all possible poses, is considered an ideal rigged mesh. Selecting desirable mesh and accurately estimating its weights is too time consuming to use in real-time applications. Rather than relying on a single ideal mesh, a set of good meshes that can animate all of the poses is created. Each mesh in this set, is capable of animating a large distinct set of poses accurately. Each of these good meshes can only accurately animate a few poses. Intuitively, the best mesh to use is a mesh that is captured in that same pose. So the set of good meshes may actually contain highly occluded meshes, like in situations where the user is standing with their arms folded for most of the animation. The selection of

good poses is dependent not only on generic factors like occlusion or deformation, but also on application specific factors, like the activities being performed or the clothes being worn during the session.

Pose selection is not commutative. Although two poses might have similar skeletons, it might take significant effort to estimate one pose from the other. For example, it might be easy to use an up-arrow pose to estimate a pose with hands behind the back. However, the reverse is not plausible; using a pose with hands behind the back to animate an up-arrow pose would yield poor and inaccurate results, primarily due to the occlusion of the hands. For this reason, a pose should not be selected based solely on its skeletal similarity.

6.4.1 Formal Definition of the Pose Selection Problem

We now provide a formal definition for the problem of optimal pose selection. We define a **Pose** as a pair comprised of both the mesh, and its corresponding skeleton $p = \{M, S\}$. The mesh selection problem is defined as follows:

Given a set of poses $P = \{p_1, p_2, ...p_n\}$, and a target pose skeleton S_T, select a pose $p = \{M, S\} \in P$ that animates the mesh M_T, corresponding to skeleton S_T, with the best visual quality.

Let us define a function $M_{out} = A(M_{in}, S_{in}, S_{out})$ that transforms a mesh M_{in} with skeleton S_{in}, into a mesh M_{out}, given the corresponding skeleton S_{out}. We use this function to redefine our problem as:

Given a set of poses $P = \{p_1, p_2, ...p_n\}$, and a target pose skeleton S_T, select a pose $p = \{M, S\} \in P$, such that the difference between a generated mesh $M_{AT} = A(M, S, S_T)$, and the actual captured mesh M_T is minimized.

The objective of this approach is to determine the optimal pose from a set of poses, so that the generated mesh M_{AT} is most similar to the actual mesh M_T. If it were possible to directly derive the difference $|M_T - M_{AT}|$, then the solution would be straightforward.

However, due to the noisy camera capture, the meshes that are generated for the exact same scene can look very different at a detailed level. Even in the absence of noise, there can be variations in vertex density, triangulation, etc., that might incorrectly manifest as errors due to animation. This kind of similarity is generally verified visually by people. The mesh quality metrics that are described in Section 6.1, predominantly concentrate on analyzing variations from the original to generate the score. In this case, the inaccurate results are due to the noisy quality of both the original and target meshes.

Therefore, we redesign our problem from one of minimizing mesh difference to minimizing transformation artifacts. The objective is to measure the amount of artifacts that are generated as a result of transforming a mesh M into a mesh M_{AT}, to correspond to the captured skeleton S_T. We refer to this as Distortion Score (σ). If the Distortion Score $\sigma(p)$ can be computed for each pose $p \in P$, with respect to the target skeleton S_T, then the optimal pose to be selected would be $argmin(\sigma(p))$.

6.4.2 Distortion Score

We define Distortion Score σ as the amount of deformation that mesh M, in pose S, has to undergo to look like mesh M_{AT} in pose S_T. Intuitively, the lesser the deformation, the lesser the artifacts created, thereby resulting in a more accurate rendering. Using mesh frames, that are in the local temporal neighborhood of the target pose, has been shown to yield accurate animations (Raghuraman et al., 2013). It has also been shown that a large amount of artifacts are generated for some frames, even in the same neighborhood. Therefore, merely selecting a pose based on the minimum effort distance does not necessarily ensure minimum artifact. As a result, skeleton similarity measurements cannot be used to estimate the σ.

All of the artifacts are modeled as excessive deformation. For this, we define deformation as the degree of change in the size of the triangles of the mesh. As mentioned in Section 6.3, meshing, rigging, segmentation, and skinning artifacts result in the excessive stretching,

or shrinking, of the triangles of the mesh. Occlusion causes holes, and clipping of the mesh, neither of which lead to any measurable deformation. The artifacts caused by self occlusion can be measured using the Distortion approach, by changing the criteria used for surface reconstruction. The 3D reconstruction approach uses the depth information to group neighboring pixels together forming triangles. The use of depth information ensures that the foreground object does not get meshed together with background objects and that the resulting mesh is largely Delaunay triangulated. Since the background is already removed from the depth image before meshing, ignoring the depth information only leads to triangles being formed between the body parts that are occluded and the parts that cover them. So when either of these body parts move, it causes the connecting triangles to stretch increasing the corresponding Distortion Score. Most of the clothing related artifacts are also captured by this reconstruction strategy. Some details, like minor wrinkles and texture mapping, are not handled by this strategy.

At an abstract level, the Distortion Score σ measures the cumulative degree of the changes in the dimensions of all of the triangles of the mesh while its being transformed to the target pose. The mesh is transformed into the target pose by displacing the vertices of the mesh. These vertex displacements cause the edges of the mesh triangles to vary. The variation in edge lengths aptly encode the impact of the transformation on the mesh. The cumulative sum of all of the edge variations provides a single value summarization of the animation process to the mesh. The holistic score, that is calculated as the difference between all of the edges of the original and deformed mesh, would cause large variations in a few of the edges to go unnoticed. To give higher precedence to large edge variations, only the edges that can be deformed are considered. In the current animation strategy, vertices can be associated with only one segment, so only the edges connecting the vertices from two different segments can get deformed. This reduces the computation cost for estimating the Distortion Score significantly, while maintaining the level of contribution of each deformable edge to the Distortion Score.

Figure 6.3. The effect of using various scoring methods: *(left to right)* selected pose with holistic scoring, selected pose with regional scoring, animated result for holistic scoring(less than 40 triangles in the head region change), animated result for regional scoring.

The mesh is divided into different regions, based on the segmentation information available from the rigging process. Edges connecting the vertices in two different regions are associated with both of them. The Distortion Score is calculated regionally by using the weighted averages of the regional Distortion Scores. As seen in Figure 6.3, by using a regional approach, the edge variations in the head region contribute more to the Distortion Score, thereby increasing it.

Given a mesh M that is transformed to M_{AT}, consisting of edges $e \in E$, regions $R = \{r_1, r_2 ... r_m\}, r_i \subset E$, then the regional Distortion Score σ_r is given by:

$$\sigma_r = \frac{\sum_{e \in r} \Delta len(e)}{\sum_{e \in r} len(e)} \tag{6.1}$$

Where $len(e)$ is the length of edge e and $\Delta len(e)$ is the change in its length due to deformation. The overall Distortion Score σ is calculated as:

$$\sigma = \frac{1}{m} \sum_{r \in R} \omega_r \sigma_r \tag{6.2}$$

where ω_r is the weight factor associated with the corresponding region of the mesh. The region based approach provides control over the influence of different regions on the overall Distortion Score for a mesh, thereby facilitating fault isolation. For example, for the mesh shown in Figure 6.3, the regional scores indicate that the regions corresponding to the hand and the head are erroneous. This also allows us to assign higher precedence to certain regions of the body over others, while computing σ for the entire mesh. For instance, a 3DTI session focusing on upper body rehabilitation can give higher ω values to the upper body, compared to the lower body. Some regions like the thumb, palm, and foot are prone to capture errors; reducing the ω values for these regions improves the result significantly.

6.4.3 Pose Selection

The optimal pose for a given target pose requires selecting the pose with the lowest σ, with respect to the target pose. The search is in the order of $O(NE)$, where N is the number of candidate meshes, and E is the number of edges in a mesh. A single camera mesh can contain hundreds of thousands of edges. So, every new addition to the pose set significantly increases the processing time for the linear search. We have considered two strategies for searching the pose repository and selecting the candidate mesh:

- Exhaustive Search: The search can be performed in parallel to calculate the σ for each mesh. However, even this strategy might result in slow selection for a large set of poses. Computing the σ for each and every pose in a large pose set and taking the minimum, may not be feasible for real time performance.

- Threshold-based Search: The search uses an approximation to select the candidate pose with acceptable visual quality. If we are able to identify the σ associated with acceptable visual quality, then the search operation only needs to find a pose that yields a σ that is less than the threshold. The computation cost of this kind of search is reduced tremendously, by foregoing the need to find the optimal mesh. The worst case scenario, of no mesh being able to provide acceptable visual quality, results in an exhaustive search and heavy computation costs. The worst case would typically only occur when the good mesh set is small, still leading to lower search times.

  For this strategy, we use the session information to estimate the value for the threshold. A 3DTI session running for few minutes requires the animation of thousands of meshes. Each animation cycle results in an optimal pose being selected from the pose set, with a minimum σ value. The average of the σ values, over multiple animation cycles, can serve as a very good approximation of the desired visual quality. The starting threshold is set using prior knowledge of the score that was gained by running other sessions. However, during the initial stages, due to the limited availability of poses, the search is usually performed on the entire pose set.

The performance of the search can be further improved by using the temporal information about the action in progress. Optimization can be performed based on the selection patterns of the poses in the pose set. For example, if certain poses are selected more often, then higher priority can be given to calculating σ for those poses, over others in the set. In Section 6.6.1, we have discussed the performance of the exhaustive and threshold-based search.

6.4.4 Animation Sequences

Selecting the optimal pose for animating every frame, without considering any temporal aspects, might lead to temporal inconsistencies. For example, consider a scenario where a

person is waving their arms, with the just the T-pose and up arrow pose in the pose set, as shown in Figure 6.4; it is highly likely that the DISPOSE method will pick either of the poses in a repetitive alternate manner, resulting in a jittery animation sequence. In this particular situation, the shoulder region of the person changes significantly between the two poses, as seen in Figure 6.4, which causes the sequence to look even worse. Since the approach only considers the σ of each candidate mesh independently, and does not measure the quality or features of the candidate poses, these kinds of situations might occur too frequently. Even though both of the poses are equally capable of animating the target skeleton, it is important from the animation sequence point of view, to hold onto one pose for as long as possible. In the above example, if the same pose was used to animate the entire sequence, then the resulting animation would be devoid of the twitching that was most likely caused by the frequent switching of selected poses. Therefore, by giving the most recently selected pose higher priority over the other poses in the pose set, it ensures that this pose is held onto until its σ exceeds the cut-off threshold (discussed in Section 6.4.3). This causes the resulting mesh animation sequences to be more consistent and visually appealing.

6.5 DISPOSE in 3DTI

Skeletal schemes (Lien et al., 2007; Raghuraman et al., 2013) for 3D Tele-Immersion deform the last received mesh, based on the current skeleton. These techniques discard the previously received meshes, and do not have any selection criteria for the mesh being transmitted. To overcome these limitations, we propose the application of the DISPOSE in 3DTI.

DISPOSE is applied in two stages, as shown in Figure 6.1. The first stage involves directly streaming the skeleton, and applying DISPOSE on the Candidate Pose Repository(CPR) at the receiver side, to render the result. The second stage involves the processing of the captured data to produce a rigged mesh, and maintaining the CPR. The meshes that need

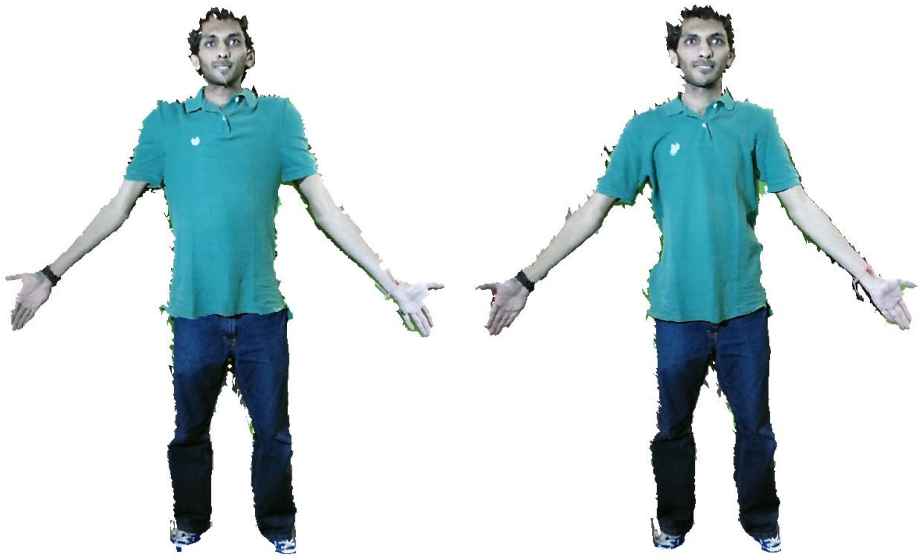

Figure 6.4. Effect of selected pose on animation: T-pose *(left)* and up arrow pose. *(right)* Due to the stretching of the shirt, while in T-Pose, the results look considerably different.

to be added into the CPR are identified at the sender side, and transmitted to the receiver side that maintains the actual CPR.

6.5.1 Sender Side - Identifying Candidate Pose

Among the meshes captured by the camera, the candidate pose mesh is identified based on its ability to animate other poses. The candidate pose mesh is identified by applying the criteria used to determine an ideal rigged mesh. The candidate pose mesh selection process used to select the mesh to be streamed from the sender side, is described as follows:

Let P_u be the universal set of all possible poses that can be performed by the person and P_w be the set of poses performed by the person in a fixed time duration. Then P_c is the candidate pose mesh for streaming in P_w, such that P_c can be used to animate a large number of poses in P_w, with a high degree of accuracy.

Candidate Pose Selection Strategy:

1. To identify P_c, the temporal neighborhood of the poses generated around P_c are used. A Usable Pose Set (UPS) stores all of the probable candidate pose meshes. A Target Skeleton Set (TSS) of the recently transmitted skeletons is also maintained. The candidate pose mesh is identified from a UPS of size N using an $\Omega(N)$ algorithm.

2. A Rigging Quotient, (χ), is calculated as N_p/N_s, where N_p is the number of meshes animated accurately (σ, lower than the selection threshold) by the pose mesh, and N_s is the total number of skeletons in the TSS.

3. When the bandwidth is available to transmit the mesh, the pose mesh with the highest rigging quotient (χ_s) is transmitted. All of the other meshes that were captured before the transmitted pose mesh, except for meshes with χ greater than Retain Threshold (RT), are removed from the UPS. RT is calculated as a percentage of χ_s, and is directly proportional to the network bandwidth.

4. All of the skeletons that occurred before the transmitted mesh are removed from TSS.

The candidate pose selection strategy ensures that only meshes, capable for animating different poses with minimal artifacts, are transmitted over the network.

Newer skeletons are added to the TSS as soon as they are detected. While adding a mesh, the σ of the mesh is compared with the existing meshes in the UPS. If the pose mesh has a low σ for an existing pose mesh, then it replaces the mesh in the UPS. Otherwise, the mesh is added to the UPS, only if it can animate at least two skeletons in the TSS with low σ. The χ for all of the meshes in the UPS are recalculated, only if the UPS has changed.

6.5.2 Candidate Pose Repository

On the receiver side, a Candidate Pose Repository (CPR) keeps track of all of the meshes received from the sender. For every skeleton received, DISPOSE is used to select the candidate pose mesh from the CPR, and animates the target mesh for the corresponding skeleton.

It is highly likely that in a high speed network environment, the sender would send a large number of meshes. As a result, the amount of time that is taken to select the candidate pose mesh, and animate the target mesh, would be substantial. So, it is extremely important to limit the size of the CPR, within the runtime bound of the system. If the animation using DISPOSE takes longer than the runtime bound, the CPR is pruned. We use an always-add-and-prune-when-required strategy for limiting the size of the CPR.

A received mesh is always added to the CPR, since the new pose mesh may have the latest facial expressions, and/or other changes in the human mesh. This addition to the CPR may require replacing an already existing candidate pose mesh, to limit the number of meshes in the CPR. The pruning strategy is as follows:

- The σ of received candidate pose meshes, with respect to all of the poses in the CPR is calculated. Any mesh in the CPR, having a σ less than the Equal Threshold(ET), is replaced by the new mesh; for example, if the received candidate pose mesh can animate one of the existing meshes very accurately, then it replaces that mesh in the CPR. ET is set to a value lower than 0.1, depending on the required pruning level.

- Each candidate pose mesh has an associated usage count, and is arranged by the order of usage. New meshes that are added to the CPR are considered as used when they are added. Also, every time a candidate pose mesh is selected, the usage counter increases for that particular mesh. While pruning, poses that have not been used or are rarely used, are removed from the CPR.

When using the always-add-and-prune-when-required strategy, DISPOSE gives good visual results in a timely manner. During our experiments, it was observed that the value of ET plays a key role in controlling the size of the CPR.

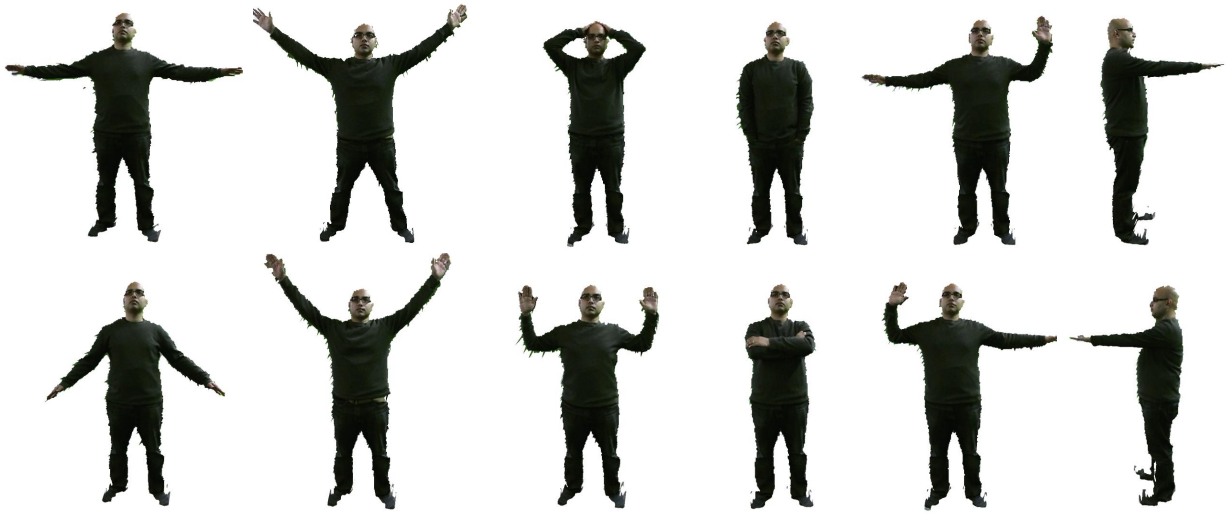

Figure 6.5. Illustration of 12 fundamental poses captured in the dataset.

6.6 Evaluation

Setup: Experiments were carried out on a system running Microsoft Windows 8.1, with an Intel i7 3.4ghz processor, 32GB RAM, and GTX 590 GPU. A single Microsoft Kinect V2 sensor was used to capture the actions of the users. Only one sensor was used to ensure that the system could compensate for all kinds of occlusion. All of the computations were performed on the CPU; i.e., a single machine was used to both store and process the information from the cameras. The entire processing pipeline used to generate the rigged mesh, as described in Section 6.2, was running on the same machine in parallel.

Dataset: A dataset of 7 participants with varying physical attributes, was captured using a single RGB-D camera, and used to evaluate the performance of the DISPOSE based 3DTI approach. A dataset of mesh streams was created by capturing users performing specific actions while primarily facing the camera. Due to the large size of data, this dataset was made available upon request. The actions involved two phases:

1. performing 12 fundamental poses, as shown in Figure 6.5, followed by

2. performing activities that the participant chose to perform for the rest of the 2 minute duration.

The actions performed in the second phase varied across the participants, with several of these actions resulting in self-occlusion (due to the use of a single Kinect camera). A total of about 25,000 frames of information were captured using the 7 participants. All of the data was used for the experiments and the user study.

6.6.1 Experiments

We carried out two categories of experiments in order to gauge the performance of the Distortion Score, and the effectiveness of using the DISPOSE strategy to generate animation sequences. The Distortion Score was calculated with all of the region weights set to 1. The threshold value was set to 0.15, for the threshold-based search.

Performance of Distortion Score

Linearity of Distortion Score: In the first experiment, the high level of accuracy, and the progressive nature of the score, are validated. The Distortion Score σ was calculated for the entire dataset, using the 12 candidate poses. Meshes were clustered based on σ, and visually inspected to verify the visual quality of each cluster. It was noticed that the quality of the meshes is progressively worse when going from $\sigma = 0$ to $\sigma = 3$; a small snapshot of the same is shown in Figure 6.6). It was possible to verify, at a coarse level, that the increase in artifacts corresponds to a similar increase in the Distortion Scores. However, at a fine level, it was very hard to determine the difference in meshes that were very close to each other, in terms of σ.

Running Time of Distortion Score: Theoretically, the running time of DISPOSE is linear to the number of meshes in the CPR; i.e., $\Omega(NE)$, where N is the number of meshes in the CPR, and E is the average number of edges in a mesh. But parallel implementation

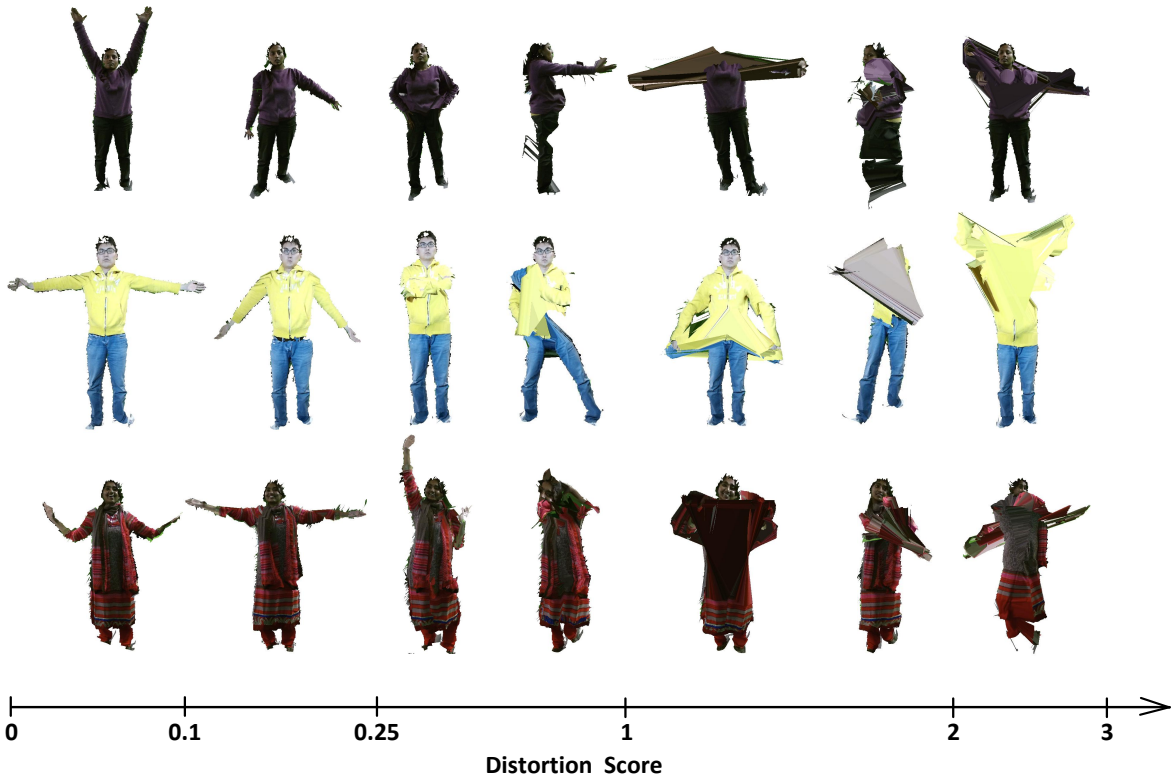

Figure 6.6. Instances of Varying Distortion Scores from multiple participants, in increasing order from left to right.

of the exhaustive search approach on the test machine resulted in non linear growth, as the number of meshes increased. As seen in Figure 6.7, after crossing a threshold of 19 candidate pose meshes, the parallel exhaustive search starts to take longer than the linear growth, and becomes infeasible to compute and use in real-time. Since the CPU was used, the performance reduction was not caused by a lack of memory as seen in the threshold search results, but was instead triggered by the large number of threads used to compute the score.

In comparison, the threshold based selection, described in Section 6.4.3, performs faster, even with a large CPR. The advantage of stopping the search after finding a reasonably good match, allows the CPR to grow to more than 100 meshes, with the result computed in under 20ms. However, the threshold-based search also takes the same time as the exhaustive

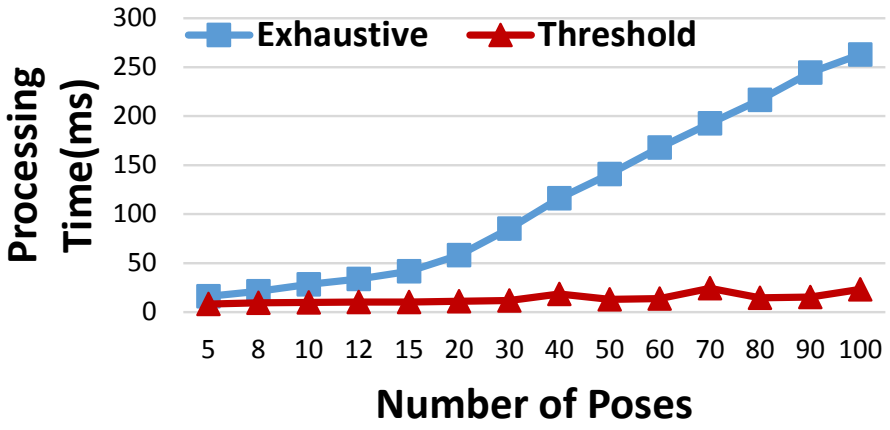

Figure 6.7. Comparison of processing times for the exhaustive search and threshold based search.

search when the quality threshold is set to very high quality, and no good candidate pose mesh is available. If the number of meshes in the CPR is pruned, the threshold-based search time could be maintained below 20ms.

Efficacy of Threshold based Search: It is clear that the threshold search approach of mesh selection is faster and more scaleable, than the exhaustive search. The qualities of the result, generated by both the exhaustive and the threshold-based approach, were found to be highly similar. To study the difference in quality, frames with the highest variation, between the threshold and the exhaustive search, were selected. The typical threshold was found to be 0.15 on the entire dataset, and used as the default starting value for the threshold. The highest variation found was 0.12, and the meshes always looked similar from the overall artifact point of view. As seen in Figure 6.8, it is very difficult to determine which of the two is of better quality, after ignoring the edge noise from RGB-D capture.

Evaluating DISPOSE in 3DTI

Latency and Frame-rates: Next, we compared the performance of DISPOSE-based 3DTI in terms of the payload size and the latency associated with the processing of the input, with the systems available in literature. Both skeleton based 3DTI and DISPOSE-based

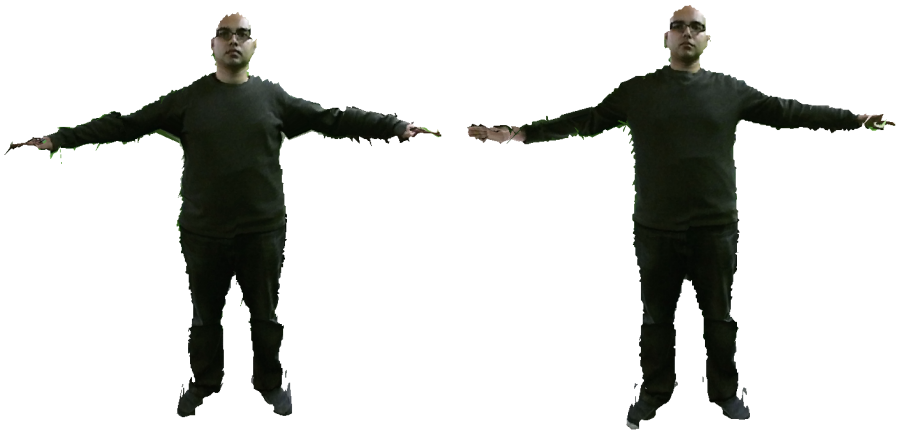

Figure 6.8. The mesh selected by the exhaustive search *(left)*, having $\sigma = 0.02$, and the threshold based selected mesh *(right)*, having $\sigma = 0.14$

.

3DTI, have the lowest payload size and highest frame rates, as shown in Table 6.1. With an average payload size of just over 250 bytes and a bit rate of about 56kbps, the results generated by the animation approaches is very responsive in real time, even over a low speed internet connection. With internet bandwidth around 1Mbps upload speed, and after an initial lead-time for a single frame transfer, the animation approaches perform at the source frame rate of 30fps; all of the compression based approaches take more time to update just one frame. The compression of (Lien et al., 2007) is comparable to our approach, for the lowest quality settings; but given its large processing times per frame (in seconds), it was not included in the table.

User Study: We performed a user study where 37 participants compared the visual quality (appearance without artifacts) of skeleton based 3DTI (Raghuraman et al., 2013), DISPOSE-based 3DTI, and the lossless compressed stream (Yang et al., 2006). The participants were shown the recorded versions of the compression based stream at 10Mbps, DISPOSE, skeleton based 3DTI, and the original stream, side by side. A total of 12 videos were generated from the dataset, with videos 7 and 9 having high self occlusion. The participants were asked to rate the visual quality and responsiveness, on a scale of 1 to 10. Due

Table 6.1. Comparison of various 3DTI transmission strategies in literature.

Method	Processing Latency (ms)	Payload Size(Mb)	FPS*
(Petit et al., 2009)	NA	29	15
(Zhou et al., 2011)	80	8	14
(Mekuria et al., 2014)	151	58	5
(Alexiadis et al., 2014)	500	7	20
(Yang et al., 2006)	159.5	NA	5
(Yang et al., 2010)	106.9	NA	7
(Alexiadis et al., 2013)	136.7	NA	7.3
(Mekuria et al., 2013)	200	4.3	8
(Kurillo and Bajcsy, 2013)	47	NA	20
(Raghuraman et al., 2013)	1	0.002	30
DISPOSE	6	0.002	30

NA: Data not available
* - as reported based on network conditions used

to the absence of formal definitions and specifications for these qualitative measures, the responses reflect the participants' respective subjective interpretations. The preferred approach of the participants, based on the choice of visual quality, is shown in Figure 6.9. Both, of the skeleton based 3DTI and DISPOSE, approaches had high ratings for responsiveness, while the compression based stream was considered the worst.

6.7 Discussion

Using the DISPOSE approach to generate animation sequences is very effective, at generating real time animation, using an online stream of meshes (no correspondence) and skeletons. We discuss a few observations of the approach below.

Ability to handle self occlusion: This experiment focused on the ability of DISPOSE to improve the capture quality of the original meshes, specifically in terms of self-occlusion. The raw images captured using the Kinect, and other pin hole based RGB-D cameras, is

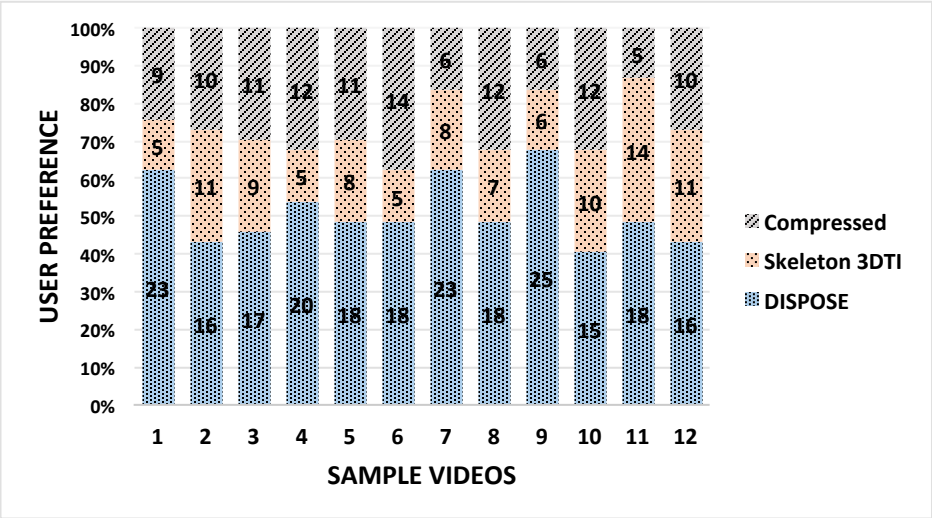

Figure 6.9. Results of the user study for preference of transmission strategies.

highly susceptible to self occlusion, as shown in Figure 6.10. Since the DISPOSE based approach chooses a mesh that will result in minimal distortion, it was noticed that the visual quality of the rendering was better at the receiver side than the sender side; this is due to portions of the mesh being missing, due to self occlusion at the senders side. Since the pipeline to capture, mesh, and render takes slightly longer than the DISPOSE based rendering, using DISPOSE also at the sender's side can increase performance considerably, especially when using multiple camera capture.

Skeleton similarity based selection: Pose mesh selection based approaches (Xu et al., 2011; Lewis et al., 2000; Hilsmann et al., 2013) rely on the similarity between the pose and the mesh skeletons in the database, to select the appropriate mesh to animate the mesh. We found that skeleton similarity based indexing, distance metric based selection, etc., work effectively in scenarios with proper meshing, rigging, segmentation, and skinning. However, online real-time applications, like 3DTI skeleton match based approaches, generate many artifacts (Figure 6.3). Skeleton based strategies are faster, and may be required to achieve higher frame rates at lower processing costs. Currently using the threshold-based search, it is possible to achieve 30fps on the CPU, for pose mesh sets with about 200 meshes.

Figure 6.10. Capture with self occlusion: *(left to right)* captured mesh with occlusion, selected pose, and animated mesh without occlusion.

Effect of inaccurate skeleton: As we mentioned earlier, we use the Microsoft SDK for extracting the skeletal data stream from the Kinect cameras. Skeletal data is quite accurate in normal exercise poses, but is prone to error in cases of occlusion. We observed that, in about 780 of the 25,000 captured frames, the skeleton associated with the mesh was inaccurate. Since DISPOSE uses just the skeleton to animate the result, all of the rendered sequences for these frames were inconsistent. In cases with heavy distortions, as shown in Figure 6.11, the frames were not rendered to the user, due to very high σ.

Unlike the animation part, the pose set update process can tolerate large skeleton inaccuracies(greater than 5cm), effectively. Larger errors result in very high σ for the mesh, resulting in low χ, thereby keeping the mesh from being added to the candidate pose set. Certain meshes, with small skeletal joint errors, can still have a low σ and a high χ, forcing its addition into the candidate pose set. Estimating a single skeleton, using multiple Kinects, has been shown to be fairly resistant to occlusion (Yeung et al., 2013).

Other distortion/quality scores: The preliminary analysis, using faster quality metrics like the dihedral angle, did not provide good quality results. Significant noise, in the

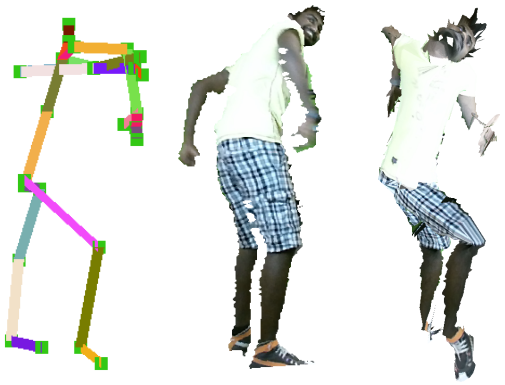

Figure 6.11. Incorrect skeleton detection, resulting in inaccurate animation: *(left to right)* detected skeleton, actual mesh, and animated mesh.

captured depth information, affects the scores determined by the quality metrics tremendously. It was often observed that the animated meshes were given a higher score than even the originally captured meshes. Given this ambiguity, no quantitative comparisons were performed using the quality scores.

Skinning artifacts: Our 3DTI system currently uses fixed weights for the entire segment. Spherical blend skinning produces joint artifacts, even on meshes rigged manually by experts. Selecting the best rendered mesh, based on these artifacts, is therefore beneficial even when multiple expertly weighted meshes are available, or in any of the pose space based approaches. DISPOSE simply selects the most suitable animated mesh, based on σ; and since it does not play any role in the actual vertex updates, the animation quality is ultimately decided by the skinning approach used to deform the meshes.

CHAPTER 7

SECURITY IMPACT OF PREDICTION

RGB-D cameras, such as Microsoft Kinect, have become very popular among computer vision researchers because of their ability to provide depth information, which reduces the complexity of some key vision problems. Hence, these cameras are being used in numerous applications, including surveillance, interactive advertising, etc. In particular, the depth image data has been used to extract a human's signature in a scene and for re-identification of the human by signature matching (Barbosa et al., 2012; Vezzani et al., 2013). RGB-D cameras produce the same depth images in a minimally illuminated scene or in the dark; normal digital cameras can not be used in this scenario. This introduces a new area of surveillance based on depth images. To the best of our knowledge, most research in using RGB-D cameras for surveillance focus mainly on using the depth image for extracting information on the human(s) present in the scene. Not much seems to have been done on the vulnerabilities of the forensic tools. On this front of vulnerability analysis, previous research (Milani et al., 2012) focuses mainly on many forensic and anti-forensics techniques for image and video manipulation, with little explorations on depth or 3D stream manipulations.

In this paper, we start by presenting an anti-forensic 3D object stream manipulation framework to capture and manipulate live RGB-D data streams to generate realistic images/videos showing individuals doing activities they did not actually perform. This anti-forensic framework takes raw live or recorded RGB-D streams and a skeleton sequence as input. The skeleton sequence can come from another live/recorded stream or one created using animation software, like Autodesk Motionbuilder. The system then produces a real time realistic sequence of 3D models, like a 3D reconstruction system would, but with the actor in the live stream performing actions shown by the skeleton sequence. To produce this stream with modified behavior, the framework identifies the actor and generates a 3D reconstruction: it detects the skeleton pose in every frame, segments the depth image of

Figure 7.1. The behavior manipulated rendering of a person

the actor, and then correspondingly deforms the 3D mesh in real time. The delivered result gives the end user the impression that the actor on the screen is performing the activity (see Figure 7.1). (This anti-forensic framework can also introduce human(s) into a scene where there were no humans). We then conduct a user study to visually inspect the manipulated depth, color and 3D video streams and check whether humans are able to identify/recognize the manipulations, just as security personnel would do. This study, using vision and graphics researchers, shows that it is indeed difficult for humans to detect the manipulations.

Next, we investigate forensic approaches for their ability to detect the manipulations. We particularly focus on depth image streams as various approaches suggest the use of RGB-D cameras for multi-attribute people in re-identification in surveillance (Barbosa et al., 2012; Vezzani et al., 2013). We use block-based depth noise evaluation approach to detect manipulations in the depth stream. The results show that inter frame noise can be a key factor in identifying certain types of depth image forgery.

Related Work: Our framework uses the depth and color stream, to produce manipulated depth, color and 3D streams. Segmentation plays a vital role in the generation of these streams. Segmentation is a widely researched topic in both computer vision and graphics. 3D methods for segmentation (Chen et al., 2009) that rely on distances, curvature, graph based cuts etc. are slow and inaccurate for human body segmentation. Methods specific for human body segmentation (Ladicky et al., 2013) rely on pixel wise classifiers or pose fitting (Huang et al., 2013) resulting in highly computational and non real time results. (Shotton et al., 2011) uses a highly optimized region based random forest to provide highly accurate skeletons using depth data. Region growing based segmentation methods are very fast (Adams and Bischof, 1994) and accurate depending on the boundary condition. Our method uses a combination of pose and region growing to achieve pixel wise real time human body segmentation.

Many methods have been proposed for generating and detecting forgery in color images/videos.(Farid, 2009; Milani et al., 2012). While anti-forensic methods focus on fooling a specific forensic algorithm, to the best of our knowledge no methods have been proposed to generate such realistic forged videos(Milani et al., 2012). Since we use depth streams, the focus is mainly on pixel based techniques that study noise and neighborhood information to detect forgery (Fridrich et al., 2003; Popescu and Farid, 2004).

7.1 Anti-Forensic Framework

The anti-forensic framework generates two types of forgeries. **Type I** forged streams place humans extracted from the real world with complex background and positions them in other complex scenes. **Type II** streams, not only extract the person, but also manipulates the behavior of the person before placing them in other scenes. Both of these forgeries are made possible due to accurate segmentation of the person and skeletal animation.

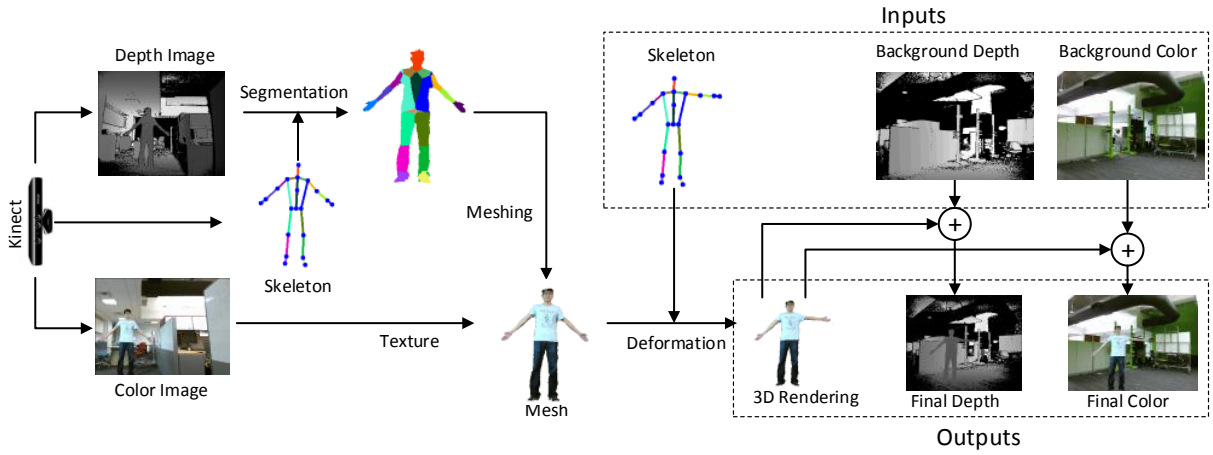

Figure 7.2. The various aspects of anti-forensic framework to generate behavior manipulated streams.

To generate the forged information, color, depth and skeletal streams of the Kinect are used. As shown in Figure 7.2, the raw values of the depth and skeleton are combined together to get the segmented depth image, with each part of the body identified. At this point the depth image only contains the person. A 3D mesh is generated using this depth image and the color image is used to add texture to this mesh. Since the depth image was already segmented, the new mesh retains the segmentation information.

The segmented mesh is then deformed using a new skeleton from a different activity. The deformed mesh is rendered in 3D to generate the processed 3D stream. To generate the color and depth streams, the vertices of the mesh are back projected from 3D to 2D and combined with a background depth and color image. Each frame in the input stream is processed using the same procedure generating a constant manipulated Type II stream. Type I stream is generated by directly using the segmented mesh. The later sections discuss the various steps in detail.

7.1.1 Real-time Segmentation

Accurately identifying and estimating the extent of various parts of the body is essential in generating good quality animation. The goal of segmentation is to accurately tag each

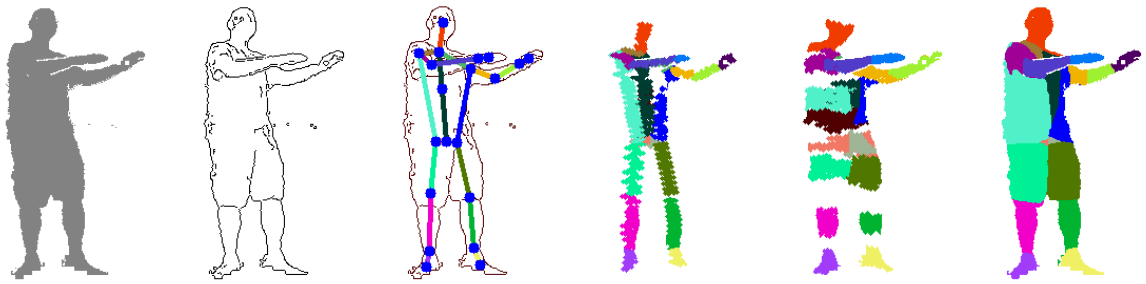

Figure 7.3. The various stages in the segmentation process – (from left to right) the depth image is filtered to extract the person, then edges are estimated, skeleton is overlaid on the edge image, segments are then identified by growing from the seed points on the joint, reaching the final result.

vertex of the body to its corresponding bone, such that any motion of the bone would result in a proportional change in the position of the vertex. A region growing based method is used to segment the depth image accurately and quickly. Various steps involved are shown in Figure 7.3 and described below:

Depth filter is applied to extract the region where the person might be present. Depth filter performs similarly to a background mask and allows the generation of depth bounds for the foreground. This method eliminates most of the background and unrelated objects from the scene.

Contour detection uses a canny edge detector on the depth image to identify the silhouette of the person. Depending on the noise levels of the depth image, some extra contours outside the body can also be detected as shown in Figure 7.3.

Skeleton identification is performed using a real time, highly accurate depth image based approach (Shotton et al., 2011) and is available via the Kinect SDK. Since only a single depth image is used for the skeleton identification, occlusion is highly likely and results in faulty skeletons. The method uses a depth image descriptor based random forest to classify and tag skeletal joints, sometimes leading to misaligned bones as shown in Figure 7.4.

Region growing is then used to perform pixel level segmentation of the image. Each pixel falling on the line between the bone joints is used as the seed point. Using the largest

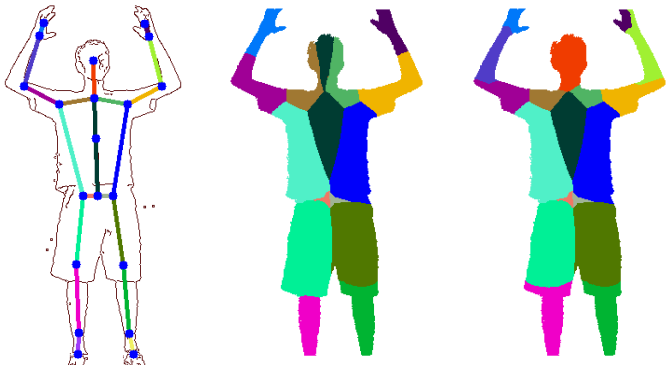

Figure 7.4. The effect of number of seeds points on the segmentation, from left to right the skeleton is misaligned a little, using a single seed point at the middle of the joint results in bad segmentation and using the multiple seeds fix gives optimal segmentation.

number of seed points ensures higher accuracy by ensuring a better coverage. The pixels neighboring the seed point are assigned the same segment tag as the seed point. The propagation of the tag is continued until either a contour pixel is reached or it reaches a pixel that is already tagged to a bone segment with shorter distance. This procedure is continued till all the outward paths from the seed points have been exhausted. The contour condition is then relaxed and all the unsegmented inner pixels are then assigned a segment using the shortest distance to the bone. After this procedure, all pixels not connected to the person remain unsegmented and are eliminated. The region growing method not only segments the person but also eliminates noisy pixels, leading to a clean extraction of the person.

7.1.2 Behavior Manipulation

Computer animation methods are used to manipulate the behavior of the person in the scene. The color and depth image are converted into a point cloud using the extrinsic parameters of the cameras. To map the texture to the points, the mapping between the color and depth image is also retained.

Mesh generation exploits the inherent structure in the depth image to quickly triangulate and produce a mesh. Comparison methods proposed in (Raghuraman et al., 2013) are

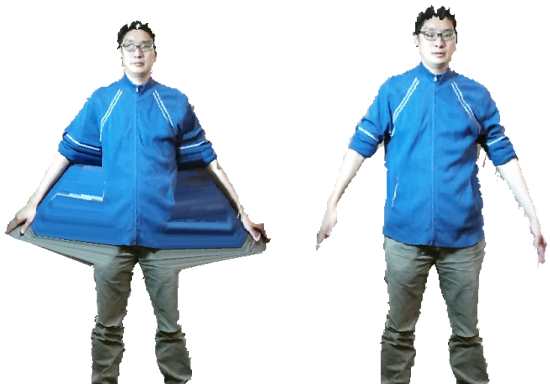

Figure 7.5. Deformed meshes generated without using segmentation information are on the left, and with segmentation information on the right.

used to identify and connect neighboring vertices. Since the mesh needs to be transformed for different body poses, the direct application method may give bad results. Consequently, the meshing method was modified to not only consider depth and spatial neighborhood, but also the segmentation information. Only those vertices from segments that are adjacent to each other are allowed to be part of a triangle. Meshes generated using this method have fewer overlapping triangles which leads to a clean skeletal animation, as shown in Figure 7.5.

Skeletal animation is performed on the segmented model using rigid body deformation. The real time streaming deformation technique described in (Raghuraman et al., 2013) is used to deform the model. In this method, each joint is represented by a control point and a representative point. Control point is the point around which the rotation of the joint happens. Spherical representation is used for all the vertices in the segment centered around the control point. The angular changes of the skeleton are first applied to the skeleton corresponding to the segmented model. Since the skeleton is generated every time for each frame using fitting, the size of the skeleton changes between frames. Due to this, all changes are executed on the same skeleton. The angles corresponding to each vertex, in the spherical representation, are then transformed by the same angular deviation as the representative point. This procedure is repeated for each new skeleton or mesh to generate the 3D object stream.

Manipulated color and depth images are generated from the 3D mesh using a point cloud representation. The 3D point cloud is generated by representing the vertices of the mesh as 3D points. The original mesh that is constructed is dense, having a large number of vertices thereby yielding a fairly dense point cloud. The images are generated by projecting each point in the point cloud to a point on a 2D plane using the Intrinsic parameter matrix of the camera. Simultaneously, a color image, of same resolution as that of depth image, is generated by assigning color values of a 3D point in the colored point cloud to its corresponding 2D point. Depth images and color images, generated in this manner, are generally noisy as some points on the 3D point cloud may not have direct corresponding points in the 2D plane. As a result, there might be holes in the depth and color images. To overcome this issue, we interpolate these values with its 4-connected neighbors, generating seamless depth and color images. The majority of artifacts in both depth and color images are eliminated using the interpolation as seen in Figure 7.6. Type I depth images are generated using the original 3D mesh, and Type II are generated after applying the deformation using a new skeletal pose.

7.2 Evaluation

To study the quality of forged streams generated by the framework, both automatic forensic methods and user study were used. All the data was captured using the new Kinect V2 sensor and was processed using a 3.4Ghz Intel CPU. The color images were captured at 1080p and depth images had a resolution of 512x424. The entire processing for generating a Type II forged stream took about 20ms. The number of seed points does not play any role in the time taken by the segmentation algorithm. The reduction in number of seed points can have an adverse effect on the accuracy of the segmentation as shown in Figure 7.4. Using standard distance based techniques, like Voronoi diagrams, for estimating segments

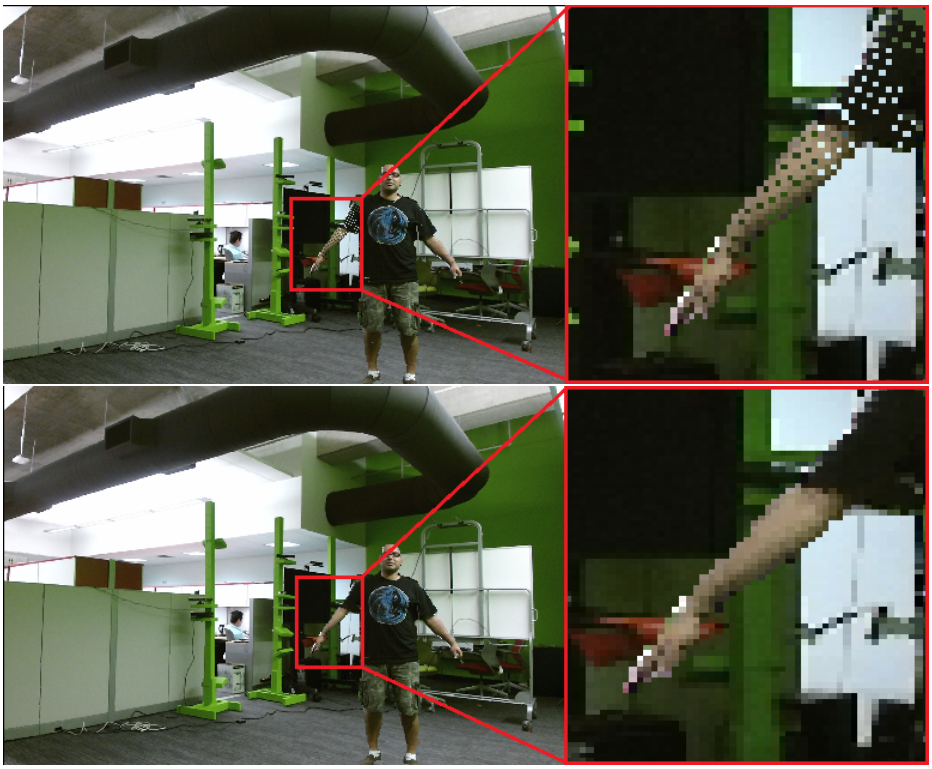

Figure 7.6. Altered color image generated without interpolation on top and with interpolation on bottom

as described in (Raghuraman et al., 2013), result in bad segmentation when the bones are close to each other as shown in Figure 7.7.

Six different people were captured performing different activities such as waving, raising left arm, raising right arm and raising both arms in three different backgrounds. All actions were performed with the subjects facing the camera and special care was taken to ensure that the actions lie on the camera plane. Using this captured raw data, we created different sets of Type I and Type II forged depth, color and 3D rendering sequences. The automatic method described below used all the depth samples, whereas a small percentage of samples was selected for the user study.

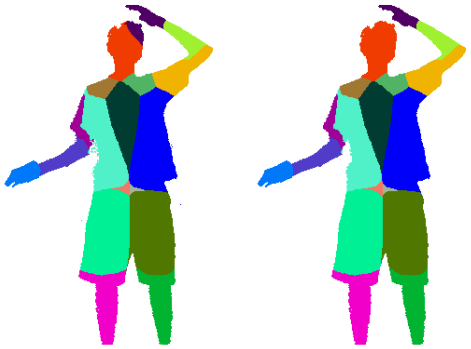

Figure 7.7. The segmentation done using Voronoi on the left and edge based method on the right, notice the head region

7.2.1 Forensic Evaluation

RGB-D cameras like Kinect generate noisy depth data. The noise levels are so high that they can be easily noticed by looking at consecutive depth frames. If the forger uses a single depth frame for the background, then the sequence generated will not exhibit this noise characteristic. Unlike color images, insertions into depth images do not influence the pixel neighborhood, but it reduces the distortion artifact present around the edges of the depth images. Insertions into the background do not cause such noisy artifacts and can also be detected. Based on these observations a noise analysis based method was developed to detect forgery.

To analyze the noise variation in original and forged depth images, we define two noise measures: intra frame noise IFN and inter frame noise $\widehat{IFN}$. First a given frame I_t at instance t is divided into a set of N_R grid regions R. Then $IFN_t = |\sum_{n=1}^{N_R} |(1-\sigma)I_t(n)|/N_R|$, where $I_t(n)$ is n^{th} region of image grid and σ is a gaussian smoothing filter of size $5X5$. The inter noise between the frames t and $t+1$ is given by $\widehat{IFN_t} = 2|IFN_t - IFN_{t+1}|/(IFN_t + IFN_{t+1})$. Using $\widehat{IFN_t}$, type I and type II depth image forgeries using a static background can be detected as verified by our experiments.

Experiments were performed on sequences of Type I and Type II forged images. We computed *intra frame* noise for each frame and *inter frame* noise for each pair of consecutive

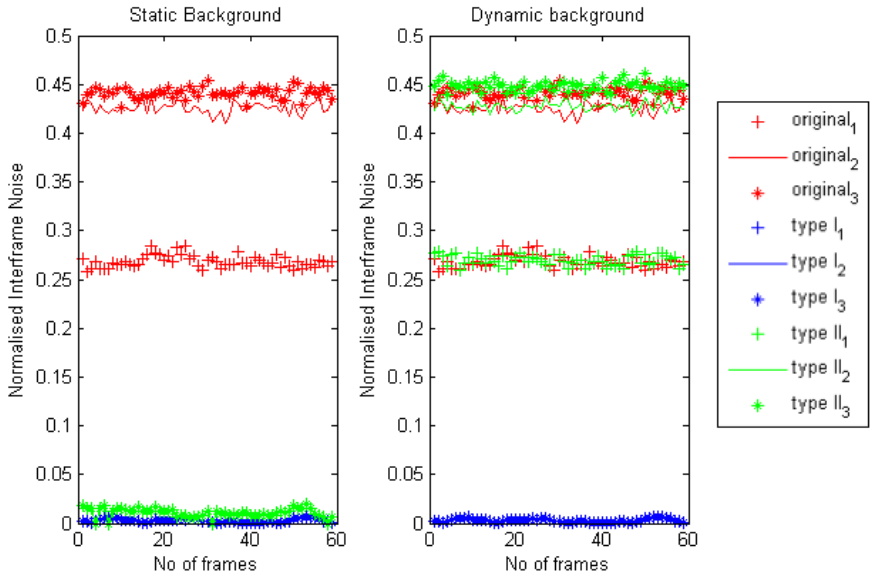

Figure 7.8. Noise analysis for original depth images, Type I and Type II forged depth images with single image and multiple images as background in all frames.

frames. Following plots assist in understanding the estimated noise ranges in all the cases. Figure 7.8 shows the plot of variation of normalized inter frame noise across a number of frames for three different datasets. From the plots, it is very clear that normalized inter frame noise in the original is very high compared to that in forged depth images; Hence it can be used as a distinguishing factor between original depth images and forged depth images.

In the second set of experiments, we generate a sequence of Type I and Type II forged images by inserting the depth image region corresponding to a person into a sequence of depth images of a background. We repeat the above experiments on the new dataset. From the noise values obtained in this case, it is clear that the simple noise based depth image forgery detection may fail to distinguish between the original depth image and the back-projected depth image. As shown in Figure 7.8, the obtained noise plots illustrate that it is difficult to distinguish between the original and back-projected depth images based on inter

frame noise as the noise level in both cases is similar. As future work, we will explore other approaches for detecting forgeries in depth images.

7.2.2 User Study

A panel of 6 graduate student researchers in computer vision and graphics were used to evaluate the results. Each student was given a set of images and video sequences of variable lengths. They had to identify if the images/videos are real or manipulated or undecidable, and also explain what identifying feature they used to make the distinction. In all, more than 100 images and 20 videos were used to evaluate the system. A few similar images and videos were provided to the participants to give them an idea of the quality. All the participants were given the same set of images/videos for evaluation. The data consisted of 40 depth(10 real, 15 Type I, 15 Type II), color 40 depth (10 real, 20 Type I, 10 Type II) and 20 3D rendered images (10 real, 10 Type II). Two videos each of real, Type I and Type II of depth, color and 3D were provided, along with 4 low resolution(240x160) color videos. The low resolution videos were provided to check on the effect of video quality on identification. Live evaluations were carried out by switching between live camera and manipulated camera feeds in real time. This dataset will be made available, and a subset of it is uploaded as supplementary material.

The participants were able to identify both Type I and Type II forged high resolution color videos. Some of the forged color images were also identified correctly. The participants mostly studied the edges of the person in the images to detect forgery in both the situations. While evaluating the low resolution color videos, all the participants were unable to identify the difference 76% of the time. The low resolution encoding of the videos introduces artifacts in the videos, which was most often the confusing aspect to the participants. The higher resolution data is easier to identify, mainly due to the noise from the depth information getting propagated onto the color image segmentation of the person. With newer and better

capture technology, these artifacts will disappear and make it much harder to identify forgery visually in color streams.

Both depth and 3D rendering proved to be very challenging tasks for the participants. With the noisy nature of the edges in the original data for both depth and 3D, there was no easy visual way for the participants to identify forgery. Similar to the automated method for identification, when the participants were given depth video in a single background depth frame, everyone could identify it as forged. However, with a sequence of background frames, confidence and accuracy of classification dropped significantly. 3D rendered Type II images/videos generated with no occlusion were always classified as original. Some were able to identify the forgeries after careful examination of the image for stretching and texture artifacts, due to the striped clothing of the person in the video. Since the other videos and images had plain clothed people with very little texture information, participants were unable to detect the forgery with confidence.

7.3 Discussion

The anti-forensic framework works very effectively while generating 3D renderings and depth streams. If only depth streams are used to process and identify like in (Barbosa et al., 2012; Vezzani et al., 2013), then it is easily possible to fool the systems, giving false results. Much more robust methods based on the characteristics of the noise around the edges need to be developed. As shown in the evaluation section, detection methods using just noise based parameters can be easily overcome.

Color images/videos are easily identifiable mainly due to the crop/add nature of the insertion into the background. Using image blending for insertion will avoid sharp edgy additions, and may result in even lower detection rate as far as users are concerned. Forgery in low resolution videos, similar to the ones generated in modern surveillance systems, is already very hard to detect for users.

In the case of 3D renderings with the constraint of no occlusion, it is clear that streams of information can be manipulated without the knowledge of the captured individual. It was possible to identify the altered stream in two scenarios – (a) when the motion induced was on a different plane, resulting in occluded or extended body parts, and (b) when the skeleton was not identified correctly for certain poses. Both the scenarios involve minor issues that can be rectified by careful frame selection. Since this study deals with real time stream manipulation, frame selection was not considered. Almost all user study participants felt that the only way to identify the difference is to look at the overall shape of the body and compare it to the actual person. Despite being intuitive, this method produced a lot of true negatives. Some even tried to do shadow analysis, but were inconclusive since the shadow is generated virtually. Many vision problems are easily solved by people, but the problem of identifying counterfeit streams of 3D information is very hard to solve even for humans. Observing anomalies of the framework remains the only way to identify manipulated streams. As technology improves, artifacts will become fewer and sparser, resulting in an indistinguishable stream.

PART III

OBSERVED LATENCY

The observed latency of an i3DTI application is the delay between an event in the real world and the corresponding virtual world rendering. All of the latencies referred to so far in this document, have been implicit latencies. Implicit latency is the time between the actual arrival of the information into the application, and the request to render the processed result. We present approaches to measure the observed latency of an i3DTI system and then analyze the impact of the latency on the quality of experience of the user engaged in an i3DTI system.

Chapter 8: The observed latency across geographically distributed locations is estimated using the approach presented here. This approach requires the i3DTI setup to capture a pattern generated independent of the system to estimate latency, so it cannot be performed while the system is in use by the user.

Chapter 9: The latency estimation approach described here is specifically designed to capture the latency at a local site. This approach uses a small strobe light and can estimate the latency of the system while the user is actively using the system.

Chapter 10: The impact of latency on the user experience is studied here using a soccer game. The question of whether the users prefer better performance, or better quality rendering, is answered by analyzing user responses to the soccer game, in multiple optimization setups.

CHAPTER 8

SCENE LATENCY

3D Tele-Immersion (3DTI) systems have a wide range of applications including 3D telepresence, remote medicine, arts, gaming, etc. (Kurillo and Bajcsy, 2013). 3DTI enables multiple users to simultaneously coexist and interact in a virtual world, using their "live" 3D models. The "live" 3D models are reconstructed in real time by capturing the user using multiple RGB-D, or stereo, cameras. Each of the users can be geographically distributed in different locations, connected to each other via the internet. Each location is referred to as a site, and consists of multiple cameras and processing machines.

The use of multiple cameras results in large amounts of data being captured and reconstructed in every single frame. Large quantities of noisy information captured by cameras, along with the complexities of 3D reconstruction, lead to processing delays. To allow faster data capture and processing, multiple machines are generally used for capture and reconstruction, as shown in Figure 8.1; This results in large volumes of information being transferred even at a single site, and these network latencies contribute to additional delays (intra-site delays). In each frame, the need for real time reconstruction of the 3D model results in a dense mesh, with hundreds of thousands of triangles. Rendering these large meshes every frame, lowers the rendering frame rate, and contributes to the delay. When multiple sites are involved, these large meshes are compressed and transmitted over the internet, leading to inter-site delays.

Latency in a 3DTI system can be broadly categorized based on when the latency was caused, such as: **(a) Capture:** Time taken to capture and send the information to the program. **(b) Processing:** Time taken to process the data. **(c) Transmission:** Time taken to move data from machine to machine within a site (intra-site delays) and across sites (inter-site). **(d) Rendering:** Time taken to draw and show the results to the user.

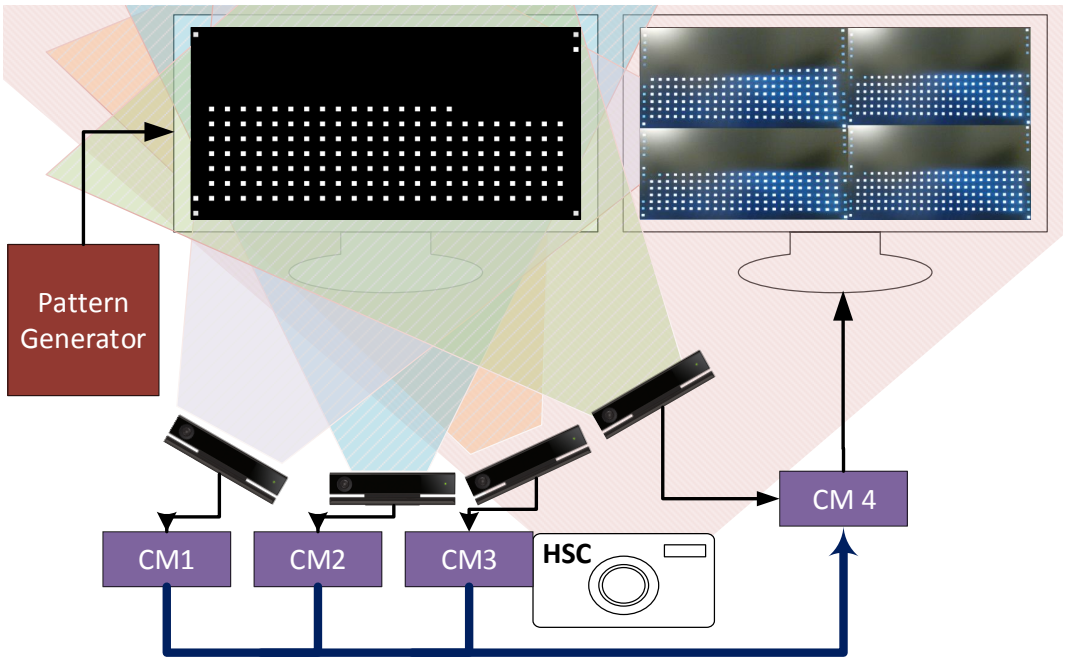

Figure 8.1. A sample layout with a high speed camera (HSC) recording the displays for the VPLE based latency estimation of a system with four Kinect V2s, processed on camera machines (CM1 to CM4).

8.0.1 Measuring Latency

Observed latency ($\mathcal{L}$) is the amount of time between a real world event, and the display of the real world event within the system. For example, if a user moves his/her arm, $\mathcal{L}$ would represent how long it takes the system to display that action. A large portion of $\mathcal{L}$ can be measured implicitly within the system. The implicit latency ($\mathcal{L}'$) is the total duration from the time the data arrived from the sensors, to the time the data is sent to the display buffer. 3DTI systems use a variety of capture and rendering technologies (both hardware and software). For these 3DTI systems, estimating $\mathcal{L}'$ involves aggregating network data transfer latencies apart from the capture, processing, and rendering latencies. Since 3DTI typically involves multiple machines over possibly heterogeneous networks, it is difficult to use clock-based mechanisms to measure these latencies. Even when clock synchronization protocols are used, the clocks may be off by twice the network propagation delay, so the

clock-based latency measurements may not reflect what is experienced by the user. In other words, no matter how detailed the implicit system delay measurements are, the $\mathcal{L}'$ might still be significantly inaccurate, compared to what is really observed.

While $\mathcal{L}'$ provides information on the delays associated with the application, it still is not the same as the delay observed by the user, $\mathcal{L}$. Most displays run at $60Hz$ and have a response rate of $5ms$, so depending on when a rendered frame is sent to the display, an extra $16.5ms$ for refresh and $5ms$ to update the screen, will be added onto the latency. There is no way to accurately measure how long it took for a rendered frame in memory to show on a screen, from within the system. Using these numbers, Ohl et al. estimated that their system would take a maximum of $30ms$ to render on a monitor grid controlled using VRUI. However, they found that it was actually taking about $80ms$ (Ohl et al., 2015). The only way to estimate $\mathcal{L}$ accurately, is by measuring it from the point of view of the user. This can only be done by having a measurement approach that is independent of the system, and is able to track both the input and output of the system from the real world.

8.0.2 Outside Observer Technique

Outside observer techniques measure latency from outside the system, relying only on the delay between the input and output of the system; Due to this, latency can be measured without the need to modify the system (in terms of the probes to be introduced in the system software). There are many approaches for the measurement of $\mathcal{L}$, for Virtual Environments (VE). VE systems do not reconstruct a real world scene but instead, rely on input trackers to regulate the rendering.

VE latency estimation methods (Steed, 2008; Mine, 1993; Di Luca, 2010; He et al., 2000) are predominantly outside observer based techniques, where an external point of reference is used to track both the real world object and the corresponding virtual rendered object. High Speed Cameras (HSC), or special sensors, are used to monitor the delay in motion between the real and virtual object, to estimate latency.

To capture a High Frame Rate (HFR) Video, the HSC needs to use low exposure rates, like 1 ms for 960 fps and 2 ms for 480 fps, videos. These low exposures cause illumination and color inaccuracies (Wu et al., 2013), leading to efforts involving manual frame counting (He et al., 2000). There are methods that are shown to work on HFR videos (Friston and Steed, 2014; Wu et al., 2013), but these rely on specific markers and need to be setup a certain way, which may not work in multiple camera setups. 3DTI systems can be geographically distributed, adding a new dimension to the problem; Replicating the same motion, with any degree of certainty in two different locations, unnecessarily complicates the problem.

8.1 VPLE

The Visual Pattern Latency Estimation (VPLE) is an outside observer technique that can measure $\mathcal{L}$ between geographically distributed 3DTI sites. VPLE is a completely independent system, and does not require any changes to the 3DTI system. The method does not require any special calibration or hardware to function accurately.

VPLE uses a rapidly changing visual pattern that encodes relative system time, instead of the motion of an object. The visual pattern is captured and rendered by the 3DTI system. The visual patterns provide a reference of time, even allowing the estimation of $\mathcal{L}$ across different locations. Along with the $\mathcal{L}$, the use of visual patterns allows VPLE to measure the system frame rate and the exposure rate of individual cameras. The generated patterns are designed to be robust against many issues, including geometric distortions, illumination changes, flickering, overexposure, etc.

An external observer (a HSC) is used to capture both the visual pattern and the rendered result of the system, to estimate $\mathcal{L}$. Videos that are captured at fixed frame rates provide very good relative time estimates. For example, each frame in a 960 fps HFR video represents 1/960 of a second (1.04 ms). The accuracy of the approach is dependent on the refresh rate of the pattern, and the capture speed of the high speed camera (HSC). A fully automated

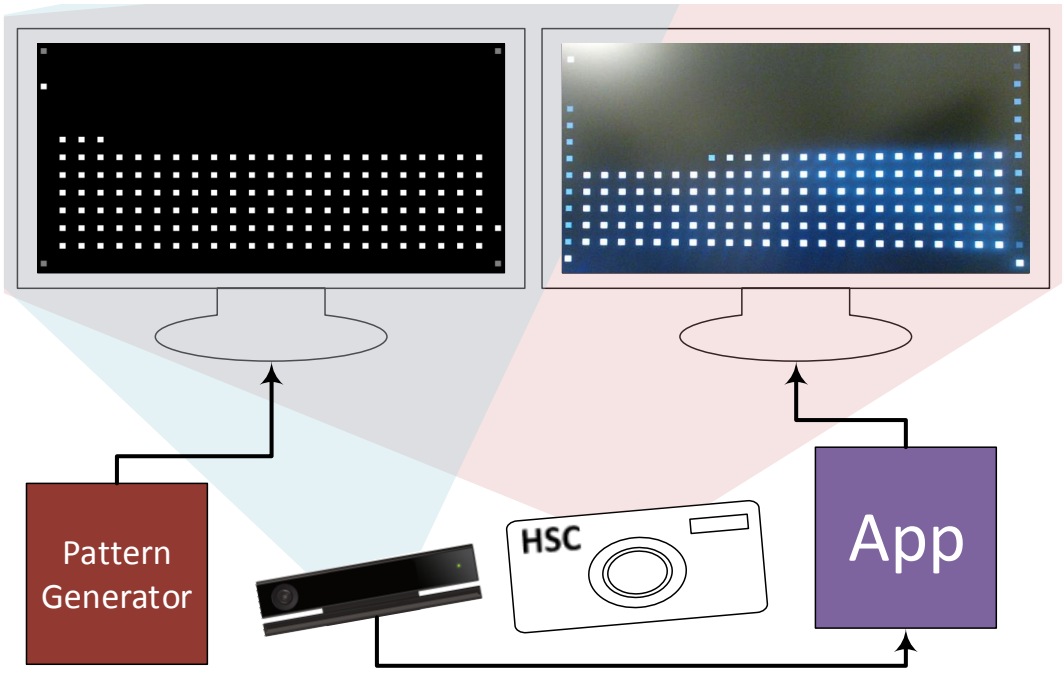

Figure 8.2. A sample layout, with a high speed camera (HSC) recording the displays for the VPLE based latency estimation application (App), with a Kinect V2.

computer vision based pattern recognition approach then decodes each of the patterns to compute latency. $\mathcal{L}$ is computed using the difference between the original and application rendered pattern.

8.1.1 The Approach

For an application with a single camera, VPLE estimation is setup as shown in Figure 8.2. The $\mathcal{L}$ of the application is estimated by VPLE as follows: A pattern generator renders a periodically changing pattern (P_G) on a display (D_G). The application camera is positioned so that it can capture the entire pattern display, along with as little background as possible. The application is then made to render the captured pattern (P_A), as clearly as possible on a display (D_A). The HSC is positioned so that it can capture both of the displays clearly. A

HFR video is recorded by the HSC, showing both the patterns P_G and P_A clearly in every frame. The HFR video is then processed offline to determine the $\mathcal{L}$.

VPLE method consists of three components: **(a) Pattern generator:** creates a unique pattern encoding the current time instance. **(b) Pattern recognizer:** decodes all the patterns from the HFR video. **(c) Latency estimator:** calculates $\mathcal{L}$ using the decoded patterns from the HFR video. These are all discussed in detail, in later sections.

8.2 Related Work

Most latency related research on 3DTI systems is primarily focused on the quality of service or the quality of experience over the network/internet (Kurillo and Bajcsy, 2013). All these works (Wu et al., 2011; Raghuraman and Prabhakaran, 2015; Desai et al., 2015; Mekuria et al., 2014; Alexiadis et al., 2014; Beck et al., 2013; Zhou et al., 2011) either compute $\mathcal{L}'$, or just use the transmission delay as the main latency. Latency has a major impact on the user's comfort in Virtual Environments (VE), leading to many techniques having been developed to measure latency in VE. Some use internal techniques, like adding hooks to Direct X API calls, to determine the exact time taken by the application to render a scene (Chen et al., 2011). A mechanical arm fitted with a rotary encoder, in conjunction with a system clock, was used by Adelstein et al. to measure the latency of trackers with different accelerations (Adelstein et al., 1996). All of these methods require modification within the system at various levels, and the last two methods require special hardware to be built.

It is not really possible to modify most VEs due to the presence of libraries and specific hardware, so most techniques use an outside observer approach. Outside observer methods track both the real and virtual worlds simultaneously, measuring the latency from the discrepancies between the two. Many techniques use an external camera to track an object in the virtual world, and a real world tracker object. Latency is computed by counting the number of frames between the movement of the real world tracker and the virtual object.

A manual frame counting approach was presented by (He et al., 2000). Steed attached a tracker to a pendulum, and tracked the virtual object and the tracker using video cameras to estimate latency (Steed, 2008). The latency estimated by the phase difference between the motion of the objects, fitted to a Sine wave. Swindells et al. measured latency by using the angular deviation between a virtual turn table positioned over the real turn table (Swindells et al., 2000). Sielhorst et al. encoded time in moving circles to measure the latency of a video see-through augmented reality system, using an external camera (Sielhorst et al., 2007). Friston et al. used a high speed camera to track a pendulum in the real world, and a virtual object using blob detectors; The latency was estimated using frame counting (Friston and Steed, 2014). Wu et al. used a 1000 fps high speed camera to track the position of a slider in the real and virtual worlds, and counted the frames to achieve close to a 1ms accuracy (Wu et al., 2013).

Other techniques use special hardware to estimate latency. Mine's method uses a photo diode to track a real world object on a pendulum, and the corresponding virtual object through a certain point (Mine, 1993). The diode, pendulum, and screen need to be positioned perfectly in order for the latency estimation to work. Di Luca used a pair of photo diodes, both looking at a gray scale gradient attached to a moving, real world and virtual world object (Di Luca, 2010); The photo diodes are connected to the sound card, allowing measurements up to 44 KHz.

While none of the methods described above can be applied directly to a geographically distributed 3DTI system, some of the methods can be modified to work for estimating latency at a single site. Ohl et al. modified (Mine, 1993) and (Steed, 2008) to measure the latency of a 3D presence system at one site (Ohl et al., 2015).

8.2.1 3D Tele-Immersion Overview

3DTI systems allow geographically distributed users to interact with each other by displaying their "live" 3D reconstructed models in the virtual world. To allow such interaction

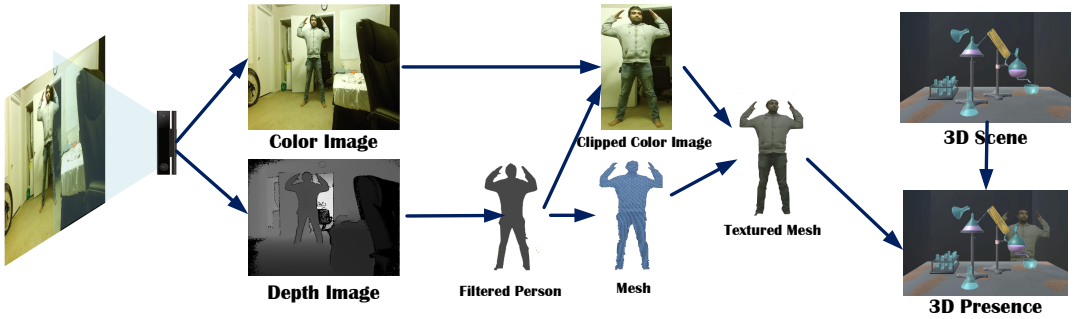

Figure 8.3. The 3DTI processing pipeline of a single site, with one RGB-D camera.

on each site, multiple calibrated RGB-D, or stereo, cameras are used to capture the user, reconstruct a 3D model, and transmit it to other sites, every frame. For 3DTI systems, that use overlapping 3D meshes from individual cameras to render the user(Kurillo and Bajcsy, 2013; Raghuraman and Prabhakaran, 2015; Mekuria et al., 2014; Desai et al., 2015; Vasudevan et al., 2011; Beck et al., 2013), every frame from each camera is processed, as shown in Figure 8.3. All 3DTI system setups in this paper use only RGB-D cameras, so the flow is described using the depth and color images as input. Stereo camera based 3DTI systems follow a very similar flow, with the depth being estimated using image disparity.

Commodity RGB-D cameras return noisy depth images, which are filtered using bilateral, median, or some similar noise removal technique, depending on the noise characteristic of the captured depth image. A combination of both color and depth images may also be used to remove noise, and enhance the depth image. After noise removal, the areas of interest can be extracted using background subtraction, region based filtering, or another equivalent method. Based on the intrinsic parameters of the camera, the depth image is projected to the real world 3D coordinate system, creating a point cloud. Due to the limited resolution of the depth image, if the generated point cloud is rendered directly, it will be full of holes. So instead of rendering a point cloud, a 3D surface mesh is reconstructed from the data. Depending on the surface reconstruction method, either the point cloud or the combination of the depth image along with a mapped point cloud, are used to generate the mesh. 3D mesh

generation is the most important, and processing intensive, step in the 3DTI pipeline. The color image is clipped and mapped to the 3D mesh, using the extrinsic calibration between the color and depth images. The meshes from each of the cameras are then realigned, based on the calibration settings between the cameras. The realigned meshes are rendered in a 3D virtual scene, using texture shaders to handle the overlap between the meshes. The texture shaders use the capture angle, surface normal, etc., to blend the colors from various mesh textures, and then render the final mesh in the scene.

In setups having multiple sites connected over the internet, the data is transmitted over and rendered on displays at the other sites. There are many techniques (Vasudevan et al., 2011; Mekuria et al., 2014; Raghuraman and Prabhakaran, 2015; Desai et al., 2015) developed specifically for the efficient transmission of 3DTI data. Evaluating these methods are beyond the scope of this work. The main factors influencing the latency of all 3DTI systems are the number of cameras used, network performance, camera capture rate, image resolution, hardware data transfer speed (camera to machine or machine to display), size of data transferred, location of object relative to camera, shape of object, data transfer rates between main memory and GPU memory, and CPU/GPU utilization and synchronization delays. So instead, the $\mathcal{L}$ of the following three commonly used 3DTI meshing approaches, along with a geometry shader based implementation, are studied:

Bisection Meshing ($3DTI_B$) (Vasudevan et al., 2011): A low poly mesh is generated from the depth image, by bisecting triangles based on the standard deviation of the region inside the triangle. The method starts by dividing the depth image into two triangles; If the standard deviation of the points inside the triangle is less than a threshold, then the triangle is added to the mesh. If the standard deviation is greater than the threshold, then the triangle is divided into two by bisecting the largest side. Whenever necessary, extra triangles are added to keep the mesh conformed. This procedure is continued till either the criteria is met or the lowest size triangle is added to the mesh.

CS3 ($3DTI_C$) (Desai et al., 2015): A low poly mesh is generated by weighting each pixel, based on its curvature in the depth image. A hessian operator is used to score each pixel in the depth image. The scores are sorted and a percentage of pixels having the highest scores are selected. Triangulation is performed, using a sweep line based image meshing approach.

Depth Image Meshing ($3DTI_D$) (Raghuraman and Prabhakaran, 2015): A dense mesh is generated, using the variation in depth as the sole criteria. A square is moved across the depth image, at each point the depth values of the pixels at the corners of the square are compared with each other. If the depth values of at least three pixels are within a certain threshold of each other, then the points are triangulated. To generate a sparser mesh using this approach, a shrunken down depth image is used as input.

Shader Meshing ($3DTI_S$): The simplicity of marching a square across the depth image to reconstruct a mesh, allows the meshing approach to be parallelised easily. Each square is processed in its own thread, and generates the necessary triangles. Rather than implement the meshing approach on the GPU using CUDA or OpenCL, which would require a hierarchical merge operation to combine all the triangles generated from each thread, the meshing approach is implemented directly on the rendering pipeline, at the geometric shader using GLSL. At each pixel in the depth image, the geometric shader generates the required triangles and renders the result, without the need for an explicit hierarchical merge operation. This approach is very efficient, as long as shadowing is disabled for the mesh, or the scene only has a few lights. Since the meshing is performed at the geometric shader level, multiple renders that are typically required for generating shadows, will cause the depth image to be meshed repeatedly, leading to performance degradation.

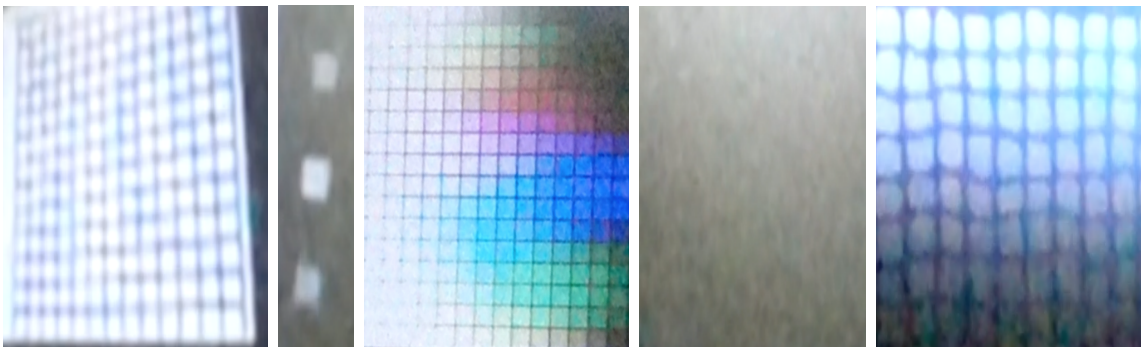

Figure 8.4. Issues with HFR, high exposure, and capture from monitors. From Left to Right: a grid captured with geometric distortion, the capture of a fast moving dot causing overexposure, the HFR capture of a color gradient grid shown on the monitor, the HFR capture of a black monitor, and flickering due to partial frame rendering.

8.3 Pattern Generation

The VPLE approach relies on the accurate decoding of the patterns captured using the HSC, from both the pattern generation display and the rendered result of the application. A new pattern is generated each time the monitor is refreshed. Each pattern that is generated depicts an instance in time. The pattern is generated in such a way that it can be detected and identified easily, from the images captured by both the high and low frame rate cameras. To enable such a pattern to be generated, we first need to identify and address the various issues involved in capturing and processing these patterns.

8.3.1 Challenges

VPLE relies on the accurate decoding of HFR video frames containing content rendered on monitors, captured using a HSC. The capture of content from monitors using cameras raises many issues; A majority of them are shown in Figure 8.4 and can be categorized based on their cause, as the following:

Geometric distortion: Using multiple cameras for capturing, it is highly likely that many of the captures are off angle, resulting in geometric distortions. Since the rendered

result of the application, that was created using the capture of the original pattern, is also captured using the HSC, each frame will now have a combination of distortions, making dense patterns undecipherable.

Overexposure: The pattern display refreshes at an extremely fast rate. Since the capture camera of the application might capture at a lower frame rate, it is very likely that any change in the luminescence of a region would not be captured by the camera, due to overexposure.

Light scatter: Even with the most precise LCD displays, there is a noticeable light scatter around the brighter pixels, that is captured by external cameras; The problem is intensified due to overexposure, which leads to distorted patterns. Reducing the brightness of the display, slightly improves the results.

Color diffusion: Since our approach relies on a high speed camera to capture both the rendered and the original pattern, the exposure rate of the camera per frame, is extremely low (just a few ms); This causes severe problems with the white balance of the image, creating illumination artifacts and washed out colors throughout the image.

Flickering: When a HSC captures a display, there is a chance of the capture happening at the exact same instance the display is refreshing; The likelihood can be reduced, by using a low response time display (1 ms) and a V-sync/H-sync rate to update the pattern.

Display Artifacts: High response rate displays refresh the screen every millisecond, but the input to these monitors is typically provided at a much lower rate. So either frames are repeated, or propriety algorithms are applied, to generate intermediate frames to reduce motion blur.

8.3.2 Visual Pattern

The visual pattern addresses all of the challenges, with the use of a binary blocks design. The pattern is laid out as a uniformly spaced grid of square blocks. Each block represents

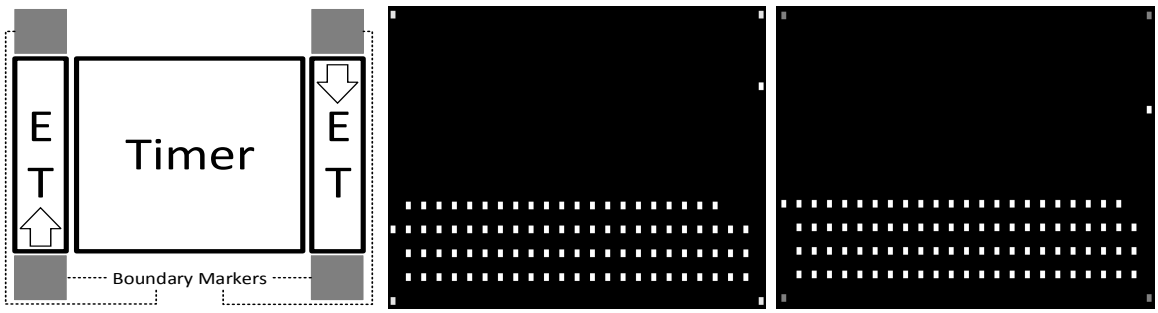

Figure 8.5. From Left to Right (not to scale): The structure of the pattern, and the sequence of 2 consecutive patterns, generated using our approach at 144 Hz.

a binary value: a solid white block indicates 1, and a black block indicates 0. The binary nature of the blocks, and the uniform spacing between each grid cell, provides the flexibility to overcome geometric distortions, light scatter, color diffusion, and display artifacts. This grid is divided into 7 regions, consisting of 3 different components, as shown in Figure 8.5.

The moment of capture is uniquely encoded into the following components:

Boundary Markers (BM): All four corner cells of the pattern grid are considered the boundary markers. The BM cells are used to represent both the size and the state of the pattern. The pattern is either completely rendered or is in the refresh state. The refresh state is caused when the pattern is captured during the refreshing of the display. To track the refresh of the display, every time a new pattern is generated, all the BM cells are either bright gray (0.9) or gray (0.5). Normally, LCD monitors raytrace from the corners in either a horizontal vertical, or a vertical horizontal manner; By having a BM in every corner, irrespective of the raytracing approach used by the monitor, incomplete display refresh can be identified. If the display is in the middle of a refresh cycle, then the BM markers would be different shades of gray.

Exposure Ticker (ET): Flickering and overexposure issues are identified using the exposure ticker. When a frame is overexposed, all the BM shades are the same high luminescence value. The ET consists of two vertical regions on either side of the pattern grid.

In each of the regions, a single active cell is moved in the vertical direction. On the left ET, the active cell moves from bottom to top and on the right side, it moves from top to bottom; This motion is repeated every time it reaches a BM. The use of opposite sides and motion directions allows the recognizer to overcome flickering. Overexposure can be identified by counting the number of active cells in each of the regions. Ideally, there should only be one active cell in each region, but in the case of overexposure, multiple cells will be active.

Timer: The timer is the most important component of the pattern. The timer encompasses the entire central region of the pattern. The pattern consists of binary blocks, so it is possible to represent the entire system clock value as the timer. However, due to overexposure, it will not be possible to accurately decode the clock from the image, despite knowing the amount of overexposure using ET. Since the aim is to measure latency, a timer relative to a specific reference in time in milliseconds (τ), is sufficient.

The timer is represented by an incremental dot pattern, to avoid any impact from all of the issues listed in Section 8.3.1. The refresh rate of the display is typically lower than the $1KHz$, so the timer representing system time in milliseconds, is an unnecessary waste of space. Instead, time is tracked using the pattern generator display refresh rate (ν_G). A counter (σ) is used to track the timer value locally, for pattern generation. At the start of the pattern, generation σ is initialized to represent the current time in milliseconds (T_C). The value of σ is calculated as:

$$\sigma = (\nu_G(T_C - \tau)) \bmod \chi \tag{8.1}$$

where χ is the maximum possible value of σ. A new pattern is generated every time the display refreshes, and σ is incremented. If σ reaches χ, σ is reset to 0. The value of χ is always one greater than the total number of cells in the timer region.

The value of σ is rendered by activating cells in the timer region. Starting from the bottom left, going from left to right, bottom to top, each cell is activated until the number

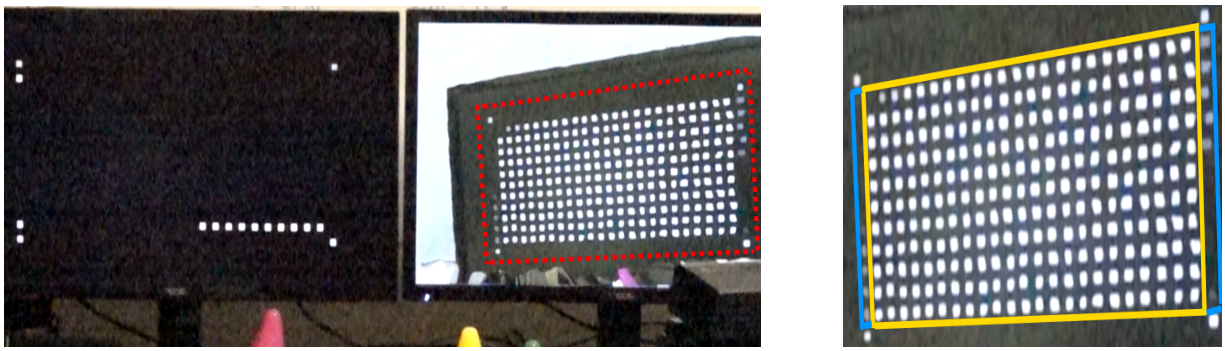

Figure 8.6. Patterns captured using the HSC from Left to Right: a generated pattern P_G, the corresponding pattern rendered by the application P_A with RoI marked in red, and the regions identified in P_A, with timer marked in yellow, and ET marked in blue.

of active cells is equal to σ; This way, the value of σ can easily be decoded by counting the number of active cells in the timer region.

Together, all of the regions allow for the automatic recognition and decoding of all the information, contained in each of the regions.

8.4 Pattern Recognition

The pattern recognition approach needs to be able to recognize the HSC captured pattern from both the pattern generator display, and the pattern on the App rendered display. While the image of the generated pattern (P_G) is clearer, given that its directly captured from the display D_G, the pattern rendered by the application (P_A) captured from display D_A, shows severe signs of the issues listed in Section 8.3.1. This is mainly due to the fact that P_A is generated by the capture, processing, and rendering of P_G by the App. A video frame from a single Kinect V2 based 3DTI setup is shown in Figure 8.6. The pattern generating process itself addresses many of these challenges, but our recognition approach still needs to be adaptable to be able to detect and decode P_A.

A single HSC frame can contain multiple patterns, depending on the scene setup. For example, a setup with multiple App cameras, as shown in Figure 8.1, can have 5 patterns

in the same frame. For this reason, the pattern recognition approach is provided with an approximate Region of Interest (RoI). For a given setup, the location of the cameras, the displays, and the rendering will mostly be the same. So, the RoI needs to be provided only once. The RoI can either be specified manually depending on the scene setup, or can be automated using computer vision techniques to detect the display. For all of the experiments in this paper, we relied on a manual RoI assessment for both P_G and P_A.

The application rendered pattern can sometimes be mirrored, depending on the RGB-D camera used. Some RGB-D cameras, like the MS Kinect, capture mirrored images. Pattern recognition is tolerant to vertical and horizontal flipped patterns because our pattern consists of symmetrically laid out components, and each component encodes data by counts.

For each RoI, the pattern recognition approach uses the following 3 stage process:

Pattern Detection: It is assumed that the given RoI largely consists of a centrally located pattern. The generated pattern is gray scaled, but each HFR video is in color; So, the region inside the RoI is extracted, and converted to a gray scale image.

The BMs represent the corners of the pattern in the image. To identify the location of the BMs, the gray scale image is divided either into a 4×4, 3×3, or 2×2 tile grid, starting with a 4×4 grid. The four corner tiles are assumed to contain one BM each. Otsu's method (Otsu, 1979) is applied on each corner tile to estimate a gray threshold for the region, based on its histogram. Since each tile is only supposed to contain black or gray regions, the threshold estimation is very accurate. A binary image is generated by applying the threshold. Independent disconnected components are identified in the binary image. The component larger than $9 pixels$ in size and closest to the corner of the original image, is considered to be the BM for the tile.

If all 4 BMs are not found, then the lower grid size is used until all 4 BMs are found. If the BMs are not found, even at the lowest grid size, then the pattern is marked to be ignored. Once the BMs are found, the mean gray level of each of the BMs is estimated.

If all of the gray level values are not close to each other (within 25), then the pattern was captured during a refresh, and the pattern is marked to be ignored. All of the ignored frames are estimated based on temporal coherence, which is discussed in the Section 8.5.

Region Detection: The pattern could have been captured off angle by the HSC, so the pattern may be distorted. This can be identified based on the location of the BMs. The pattern is a largely sparse structure, and the decoding of each component does not depend on precise positioning; So the regions are identified without correcting the distortion.

The BM regions $\{BM_{TL}, BM_{TR}, BM_{BL}, BM_{BR}\}$ located at top left, top right, bottom left and bottom right respectively, are used to determine the three remaining regions. The original pattern regions are positioned relative to the BMs, as shown in Figure 8.5. So the left ET is between BM_{TL} and BM_{BL}, and the right ET is between BM_{TR} and BM_{BR}. The timer region is between all four BMs.

The exact boundaries for these regions are drawn by using half the size of the BM as padding on each of the sides, as shown in Figure 8.6. Due to light scatter, overexposure, and geometric distortions, there is a likelihood that the cells have changed in size. The extra padding, provided on all the sides of the regions, ensures that the entirety of the cell is in the region.

Pattern Decoding: Along with other factors, the brightness of the monitor used to display the pattern, determines the level of darkness associated with black and the level of intensity of the white. For each identified region, Otsu's method (Otsu, 1979) is applied to estimate a gray threshold for each region, based on its histogram. There is a likelihood that the timer region might not have any active cells; In such a situation, the threshold returned by Otsu's method might be extremely low (< 0.2). To avoid errors in decoding, the threshold is then set to middle (0.5). The binary image is then created for each region by applying the threshold. The binary image is broken down into independent disconnected components; Components smaller than $9 pixels$ are removed, and the remaining components highlight all the cells that are active in each region.

For the timer region, the σ value is the total number of components in the timer region. The number of frames of exposure contained in the ET section is the average of the total number of components in both the left and right regions of ET. In the case where the left and right ET values are not the same, it means the screen was being refreshed at the time of capture. The exposure value is retained as calculated.

8.5 Latency Estimation

The latency is estimated by using the entire video containing both the P_G and P_A, clearly visible on each frame. For a video consisting of f frames, containing generated patterns $({P_G}^1, ...{P_G}^f)$ and application patterns $({P_A}^1, ...{P_A}^f)$, each frame is decoded to produce σ values $({\sigma_G}^1, ...{\sigma_G}^f)$, $({\sigma_A}^1, ...{\sigma_A}^f)$ from the timer region, and exposure values $({E_G}^1, ...{E_G}^f)$ and $({E_A}^1, ...{E_A}^f)$ from the ET. If any of the P_G have an exposure greater than 1, then the frame is ignored. It is highly likely that due to the low frame rate of the RGB-D cameras, P_A will be filled with high exposure values; This allows us to examine the actual instance when the App camera started the capture, and the instance when the capture was completed.

Ignored frames: The frame previous to the ignored frame is used to compute the values for E, C. If the previous frame does not have any valid values, then the current frame is also marked invalid. If more than α frames are marked invalid, then the video is considered invalid and is not used for the latency estimation.

The latency for the application is estimated by using the σ values, along with the capture frequency of HSC ν_{HSC}, and the pattern generation frequency ν_G. Latency is estimated for all the valid frames, following the first change in the σ_A value. For any frame i, if ${\sigma_A}^{i-1} > {\sigma_A}^i$ then it means that the σ was reset during the capture of the frame by the application camera, and all the next frames with the same value as ${\sigma_A}^i$ are ignored for the latency estimation.

Latency, at a frame i, is given by:

$$\mathcal{L}_i = \begin{cases} \frac{\chi + {\sigma_G}^i - {\sigma_A}^i}{\nu_G} + \frac{\eta_i}{\nu_{HSC}} & \text{If } {\sigma_G}^i + \frac{\eta_i \nu_G}{\nu_{HSC}} < {\sigma_A}^i \\ \frac{{\sigma_G}^i - {\sigma_A}^i}{\nu_G} + \frac{\eta_i}{\nu_{HSC}} & \text{Otherwise} \end{cases} \tag{8.2}$$

where η_i is the number of frames since the value of σ_G changed. Even though this situation is highly unlikely, invalid frames are treated as no change, and are included in the count of η_i.

Considering that the HSC always captures at a higher frame rate than the App camera, it is only possible for the value of ${\sigma_A}^i > {\sigma_G}^i$, if the counter has reset or if the ${P_G}^i$ was invalid. Since invalid frames are repeated using the previous frame, it is likely that ${\sigma_G}^i$ might be low due to repetitions; So by adding the remaining counter value, calculated using the frame counting from HSC, ${\sigma_G}^i$ is adjusted to allow accurate $\mathcal{L}_i$ estimations, even for minute latency situations. If the σ_G has reset, then adding the max counter value χ will provide an accurate $\mathcal{L}_i$. While measuring the latency, the instance when the capture was completed is considered; However, if the latency from the start of capture needs to be estimated, then $E_A \nu_G$ can be added to the estimated latency.

8.6 VPLE in 3DTI

3DTI systems can be setup in various different configurations, depending on the application. Examining all of these setups is beyond the scope of this paper, so the usage of VPLE for all of the base cases, from which any 3DTI arrangement can be derived, are described below:

8.6.1 Single Camera

For a 3DTI system, with a single site having one camera, VPLE can be applied directly, as shown in Figure 8.2. The pattern generator display (D_G) is placed in view of the camera, and the 3D mesh of the D_G is rendered by the system on the application display (D_A), located in such a way that both the displays can be captured together by an HSC.

8.6.2 Multiple Cameras

All of the cameras in the scene are aligned to view the D_G, as shown in Figure 8.1. Typically, all of the cameras in each of the sites are calibrated to allow a complete 3D reconstruction of the user. Since the goal is to measure latency $\mathcal{L}$, it is important to view as many camera views as possible, so all of the cameras are calibrated to render a view of the D_G, side by side on the D_A. The HSC is positioned to capture both the D_G and the D_A, placed near each other. Due to the presence and the positioning of multiple cameras, the HSC might need to be placed at suboptimal angles. All of the patterns on D_G and D_A need to be clearly visible to the HSC. The $\mathcal{L}$ is estimated from the point of view of each camera, by providing the relevant region of interest for the rendering associated with the camera, in the HSC video. Even in situations with synchronized camera capture, depending on the processing, rendering pipeline, etc., the $\mathcal{L}$s for each camera might differ. To compute the $\mathcal{L}$ for the entire multiple camera 3DTI system, an average/max of individual camera $\mathcal{L}$s can be used.

8.6.3 Multiple Sites

A multiple site 3DTI setup, as shown in Figure 8.7, consists of a display and one or more RGB-D cameras at each location. The $\mathcal{L}$ on each site can be measured using the single or multiple camera approach described above. To maintain a frame of reference between both sites, the system clocks for all of the machines are always synchronized. The clock synchronization allows the pattern generator to display the same pattern on multiple sites, without the need for the generators to communicate with each other. For this to work correctly, the pattern generators should have the same τ value. More details on the effectiveness of synchronized pattern generation can be found in Section 8.8.

Since the pattern generators on all the sites are synchronized, even the multiple site $\mathcal{L}$ estimation can be performed similar to Section 8.6.2. At every site, the meshes from the sites can be rendered side by side, to show the pattern on the D_A. Then, both D_A and D_G

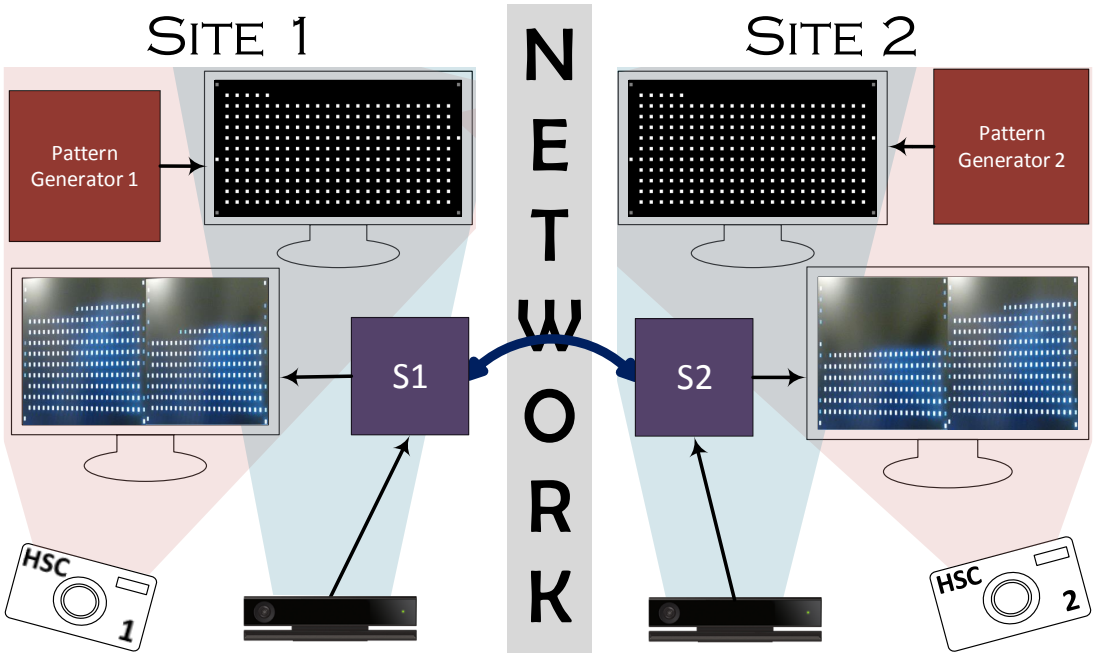

Figure 8.7. A sample setup for latency estimation, in a multiple site 3DTI application using VPLE, by capturing the displays using high speed cameras (HSC).

are captured together using the HSC. The $\mathcal{L}$ associated with the communication between the sites can be measured directly by comparing the pattern from that site, with the generated pattern.

This approach assumes that the system clocks, on all the sites, are synchronized to a high degree of accuracy. The $\mathcal{L}$ estimation can have a maximum error of up to $\pm(\frac{1}{\nu_G} + \Delta)$, where Δ is the maximum difference in the clocks of the two sites.

8.7 Implementation

A 3DTI setup consists of scores of different types of hardware. For convenience, all of the hardware used for the experiments are grouped and tagged based on their configuration. Many machines having two different configurations were used for all of the experiments: A1: Intel i7 6600k @ 4.5 GHz, 32GB DDR4 RAM, NVIDIA GTX 980 TI, Windows 8.1; and A2: Intel i5 5600 @ 3.2 GHz, 16GB DDR4 RAM, NVIDIA GTX 1060, Windows 10.

All machines were connected using a 1 Gbps wired network on a 10GbE cisco switch. Multiple monitors of the same type were used: Acer 24in LED, 144 Hz, 1ms; The following three types of RGB-D cameras were used with multiple resolutions and frame rates: C1: MS Kinect XBOX 360, with MS Kinect SDK v1.7; C2: MS Kinect XBOX One, with MS Kinect SDK v2.0; and C3: Creative senz3D, with Intel Perceptual Computing SDK. Sony RX10 III capable of capturing HFR (High Frame Rate) at a maximum of 960 fps at 1080p was the HSC used for all VPLE. The camera is only capable of capturing about 2000 frames at a time, so for each experiment, multiple videos were recorded. Since the maximum possible refresh rate for the monitors that we used was 144 Hz, all the videos, unless specified otherwise, were captured at 480 fps.

The 3DTI software is developed in C++ using CUDA 8, BOOST, CGAL, Eigen, GLEW, GLFW, GLM, OpenCV, OpenGL, Speex, TBB, Zlib and a host of device specific libraries for MS Windows. The 3DTI framework integrates with the Unity3D gaming engine to provide real-time detailed interactive scenes. Network Time Protocol (NTP) is used to synchronize the clocks of all the machines to be millisecond precise. The pattern generator is written in C++ using GLFW and OpenGL 4.4, and is capable of running at 1200 Hz on A1; But due to the limited refresh rates of the monitors, the pattern was updated and rendered at a V-Sync rate dependent on the monitor's refresh rate. The pattern generator was run on A1 and A2, rendering on monitors at 144 Hz. The pattern recognition and latency estimation from the HFR videos are done in MATLAB.

For all of the evaluations, a timer region of size 23×21 with $\chi = 254$ was used. A square pattern allowed for better detection from all angles. For situations when pattern recognition fails, the video ignore level α was set to 40 frames (2% of the total frames in each video). The α value was used as an indicator, to check if manual intervention is required.

8.8 Evaluation

To measure latency accurately, VPLE relies on many independent components. Different evaluations are performed to verify each of these components, and the entire VPLE setup.

HSC frame rate: To ensure that the HFR video captured using the HSC is always at a constant frame rate, hundreds of videos of patterns generated and displayed at $120Hz$, were recorded at 240, 480, and 960 fps, for a fixed duration of $4s$. The total number of frames in every video was always found to be consistent with the fps option. To ensure accurate spacing between the frames, the system time was logged for every 100 patterns generated. It was found that over a period of 20,000 frames, the refresh rate drifted by about 0.97%. When the patterns in the HFR video were decoded, the pattern change (σ value change) was found to be approx 1.9 frames for 240 fps, about 3.7 frames for 480 fps, and 7.67 frames for 960 fps.

Pattern Generation: The pattern is supposed to be generated at the same frequency as the refresh rate of the display. To verify the rate of pattern generation, the refresh rate for the monitor was set to 50, 60, 100, 120 and 144 Hz; The HSC then captured videos, with the pattern generator rendering the pattern on the monitor. There was a significant amount of drift noticed when the pattern generator was run inside a window vs when the pattern generator was rendered in full screen. This forced us to always run the pattern generator in full screen, for all of the experiments. The patterns were decoded from each of the videos, and the rate of growth of σ values were compared. The σ growth rate for each of the various monitor refresh rates was as expected, as shown in Figure 8.8.

Synchronized Pattern Generation: The multiple site VPLE approach relies on the synchronization between the pattern generators, to estimate $\mathcal{L}$. The variation of σ values between two pattern generators with the same τ is studied by running them on the same machine (A1) and then on two different machines (A1 and A2). The clocks on all of the machines were synchronized with a server that had a round trip time of $7ms$. The patterns

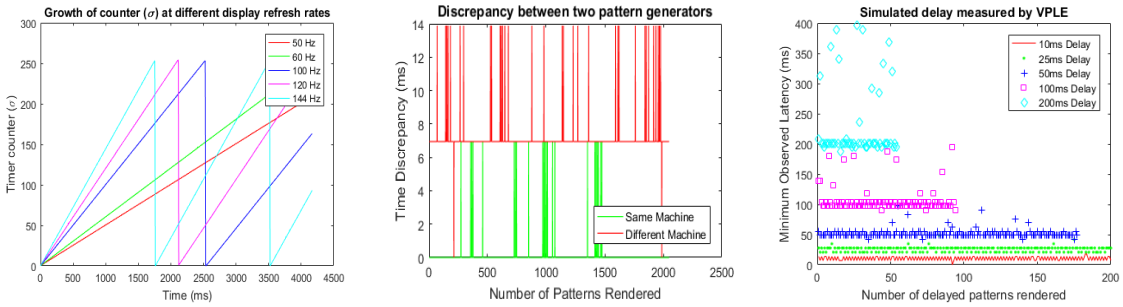

Figure 8.8. Left to Right: Variation in σ growth based on display refresh rates, mismatch between σ of two synchronized pattern generators, and VPLE estimated $\mathcal{L}$ for simulated delays.

were rendered on different monitors at $144Hz$. HSC captured both the monitors at 480 fps, and the difference in the time (ms) between the two is shown in Figure 8.8. The time deviation is only based on the difference in σ values per frame. On the same machine, the maximum difference in σ was 1, which is happening in 0.07% of the frames. The difference between the σ values on different machines was almost always at 1, but going as high as 2. It was noticed that the patterns eventually start to drift away from each other over time, and the drifts can be heavier depending on the hardware and resource usage of the machine.

Pattern Recognition: The pattern was captured directly using the HSC from various angles off the monitor. The camera was always zoomed into the monitor, to ensure that the pattern was clearly visible in the video. The pattern recognition worked accurately for captured angles relative to the monitor, between $-60°$ to $+60°$.

While the pattern recognition approach is robust against the location and the angle of capture from the HSC, the angle of capture from the application's camera impacts the recognition accuracy significantly. Since 3DTI approaches generate a mesh, which is rendered on the display, and then captured by the HSC to produce the final HFR video, any significant off angle captures of the display by the RGB-D camera leads to both noisy depth and color images. The mesh generated using this noisy data contains artifacts that lead to poor

recognition. For this reason, in all of the experiments, greater emphasis was given to the position of the RGB-D camera, over the location of the HSC.

Latency Estimation: To verify the accuracy of the VPLE approach in estimating $\mathcal{L}$, a simulation was created to add a specific amount of delay before displaying the pattern. A pattern generator (PG) was run on one machine (A1), and a display program (DP) was run on another machine (A2), which was provided with the exact σ value being rendered by the pattern generator at that instance. Once DP received a σ value, it slept for a fixed duration without receiving any other information; On waking up, it rendered the pattern with the σ value. Two machines were used to allow the patterns to be displayed in full screen, and to avoid drifts in the refresh rate due to the operating system. The baseline latency between PG and DP, without any delay, was visually undetectable at 1ms (at most). For simulated delays of $(10, 25, 50, 100, 200)$ ms, the average estimated delays were $(10.6, 25.7, 51.1, 107.2, 234.3)$ ms respectively. VPLE measures the current $\mathcal{L}$ at any given instance of time. The estimated delay is the latency at the time of DP rendering a new pattern, which is also the minimum observed latency for that data point (Figure 8.8). When DP sleeps, the packets are dropped, resulting in a reduced frame rate. This adversely effects the current $\mathcal{L}$ leading to a high average $\mathcal{L}$; In this simulation, the average $\mathcal{L}$ was found to be $(12.1, 34.2, 71.4, 147, 295.9)$ ms for $(10, 25, 50, 100, 200)$ ms delays, respectively.

8.9 Experiments

VPLE was applied to measure the observed latency for three fundamental 3DTI arrangements, to understand the latency associated with just capturing and rendering the data from the RGB-D cameras used by the 3DTI system. To decide on which RGB-D camera to use, the capture latency for each of the cameras was estimated using VPLE. For this analysis, the sample application from the manufacturers SDK was used. A well lit scene containing

multiple color objects, along with the pattern generator display, were used. The machine A1 was used to acquire, process, and render the results.

All of the 3 cameras, the Kinect V1 (C1), Kinect V2 (C2), and Creative senz3D (C3) had similar average $\mathcal{L}$ of (63.8, 65.5, 64)ms respectively. All of the cameras had similar exposure rates of 28ms. Considering the similar latencies, the Kinect V2 was selected to be used in all of the 3DTI experiments, due to its superior image quality.

In low light conditions, all of the cameras had an excess of white balance, which resulted in a highly illuminated pattern with little to no black levels. The exposure rate of only the Kinect V2 doubled, and the frame rate fell to 15 fps, which increased the latency. The other cameras performed the same as they did in normal lighting conditions. The excessive bright pixels in the captured HFR video couldn't be processed by our system, so we had to rely on using a manual threshold to extract the active cells. Given the issues that low light conditions presented, all of the 3DTI experiments were done in well lit conditions.

All 4 different reconstruction methods, described in Section 8.2.1, were evaluated in various 3DTI arrangements. The system was run for approximately 4 minutes for each of the approaches. A total of $40s$ of HSC recording was used for the VPLE. During the recorded time, the implicit latency was also measured on a per frame basis. The implicit latency was calculated by probing the system clock at the point of capture, and the point of rendering. Many of the approaches, like the $3DTI_B$, are optimized for flat surfaces and so, instead of making the cameras point directly to the display, a larger scene with a few objects was captured for all of the setups.

Single camera 3DTI: The 3DTI application was run on machine (A1). The variation in $\mathcal{L}'$ for the session is shown in Figure 8.9. The $3DTI_C$ approach took the longest duration and had a very erratic performance, with over 300ms of fluctuations (50% of the average $\mathcal{L}'$). One of these fluctuations was caught by the VPLE approach. However, the variation was not noticed in the size of meshes generated by $3DTI_C$. The excessively high $\mathcal{L}$ and $\mathcal{L}'$ (about

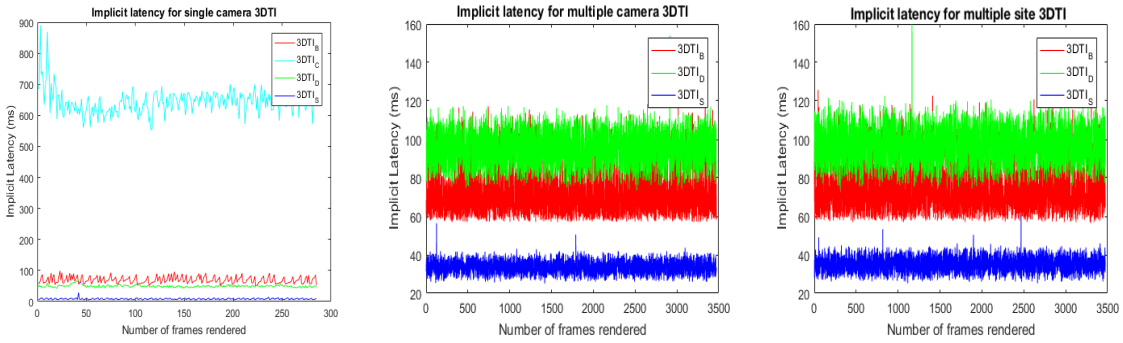

Figure 8.9. Implicit latencies for (from Left to Right): a single camera 3DTI, a multi-camera 3DTI and a multi-site 3DTI.

6 times more than all of the other approaches) lead to a frame rate of 1 to 2 fps. $3DTI_B$ approach produced meshes of similar size to the $3DTI_C$ approach (about 5k triangles) and had an average $\mathcal{L}'$ of 53.5ms, compared to 644ms of $3DTI_C$. For these reasons, the $3DTI_C$ approach was abandoned for the other 3DTI arrangements.

Both the $3DTI_B$ and $3DTI_D$ approaches had similar average $\mathcal{L}$ values of 117ms and 128ms, respectively. But according to $\mathcal{L}'$, $3DTI_D$ is, on average, 6ms faster than $3DTI_B$. The slower overall performance by $3DTI_D$ is largely attributed to the fact that the meshes generated by $3DTI_D$ are denser, with over 80k triangles. $3DTI_S$ is the fastest approach of all of the other approaches studied, in both $\mathcal{L}$ and $\mathcal{L}'$. Despite the high latency values, all of the 3 approaches ran at 30 fps, the capture frame rate of the Kinect V2.

Multi-camera 3DTI: The 3DTI application was run on machine (A1), and another Kinect V2 was connected to machine (A2) capturing the same scene. A high speed network was used to connect A1 and A2. There was no significant impact on $\mathcal{L}'$ or $\mathcal{L}$ for the local camera at A1, compared to the single camera 3DTI setup. On average, the $\mathcal{L}$ increased by 6ms, 12ms, and 4ms for $3DTI_B$, $3DTI_D$, and $3DTI_S$ respectively; However the $\mathcal{L}'$ for the local camera changed significantly, by about 22ms, 49ms, 27ms for $3DTI_B$, $3DTI_D$, and $3DTI_S$ respectively. The larger increase seen by the $3DTI_D$ approach is attributed to the fact that more data needs to be transferred and rendered (dense mesh). The image data

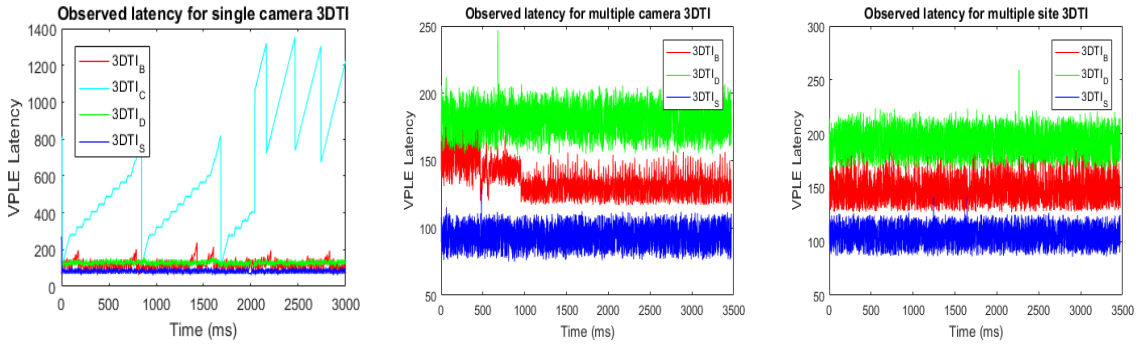

Figure 8.10. Observed latencies for (from Left to Right): a single camera 3DTI, a multi-camera 3DTI and a multi-site 3DTI.

and sparse mesh data for $3DTI_B$ and $3DTI_S$ were a comparatively smaller size, so lesser effort was required to render the data. The average $\mathcal{L}'$ for $3DTI_B$, $3DTI_D$, and $3DTI_S$ was 70.7ms, 95.2ms, and 33.7ms, respectively. Due to the network speed, there was only a modest increase in $\mathcal{L}'$ compared to the single camera setup. For the remote camera, $3DTI_D$ takes the longest time due to its larger data size. The $\mathcal{L}$ values are proportionately higher than $\mathcal{L}'$, similar to the variation in $\mathcal{L}$ for the local camera. The largest component of latency in this setup was due to the network; The capture and rendering aspects were about the same for both the single and multi camera setups.

Multi-site 3DTI: Two sites were setup with machines A1 and A2, with one camera on each site. Two other A1 machines were used to generate the pattern for each site. The sites were setup locally with high speed LAN. The system clocks were synchronized to about 3ms to each other. The τ value was set to an hour before capture. For both the local and remote cameras, the results were similar to the multi-camera setup, as shown in Figure 8.9, and Figure 8.10. Since only one camera is used on each site, on the same network, and the machines are connected the same way as in the multi-camera setup, the result is similar. The only difference between the single and multi-camera setups, is that the data is transmitted in both directions and rendered on both sites. Because of high network bandwidth, doubling the data transmission did not seem to have any significant effects on latency.

8.10 Discussion

Here we discuss many aspects of the VPLE that influence the overall usefulness of the approach and its possible applications.

Visual application: The VPLE approach can be applied to any real time visual application to measure the latency. In Section 8.9, the latency of a live 2D video feed from the cameras is measured. Since the pattern is generated at the peak refresh rate of the monitor, any application that requires an accurate visual counter can rely on the pattern generation. Using a HSC at 960 fps, provides a resolution of close to 1ms, but is not really necessary for most evaluations. The majority of the displays run at 60 Hz, so any camera capturing at 120 fps or more would be sufficient to estimate latency at acceptable levels, for applications running on those displays.

Pattern size: The number of grid cells, size of each grid cell, and the amount of spacing between cells from the HSC's point of view are the most important parameters needed to ensure the accurate functioning of VPLE. Increasing the number of cells will allow for longer periods of latency estimation, without the need to ignore certain frames due to counter rollover. The size of each cell from the HSC's video is very important because smaller size cells can cause the pattern recognition to fail. It was empirically determined that, a cell area of less than 16 pixels on a HSC captured pattern, increases the number of ignored frames dramatically, resulting in no visual latency estimation for large portions of the video.

Measurement accuracy: The measurement accuracy of the VPLE approach is highly dependent on the refresh rate of the display used to render the generated pattern. It is always possible to get a reasonable latency estimate in between refresh cycles, using the temporal coherence between the captured video frames. While this estimate can accurately capture the display update, it cannot however be used to ascertain the exact instance that (interval of $< 7ms$) the rendered pattern was captured. The experiments in this paper use a D_G with a maximum refresh rate of $144Hz$, so the minimum resolution for accurately

estimating latency is $7ms$. Monitors with refresh rates of $240Hz$ are already available, that can reduce the minimum resolution to $4ms$.

Multiple cameras: In theory, there are no real restrictions on the number of simultaneous camera latencies that can be estimated, as long as pattern detection is possible. However, in the real world, based on the size of the monitors and the distance of the HSC from the displays, it may not be possible to track more than 4 cameras at the same time. Situations involving a large number of cameras, visual latencies can be measured by disabling any view based optimization, and switching through different camera views during the HSC capture. Since the HFR video recording is manual, cameras feeds shown on the display can be changed and recorded.

IR cameras: VPLE, for the most part, uses binary patterns to determine the visual latency of a system. It can easily be adapted to measure the visual latency of an application using an IR camera.

CHAPTER 9

ACTION LATENCY

3D Presence systems capture, reconstruct, and render the user in real time. The complexity of the operations, and the large volumes of data, introduce latencies. Current state-of-the-art 3D Presence and 3D Tele-immersion (3DTI) applications rely only on the implicit latency to estimate the overall experience by the user (Kurillo and Bajcsy, 2013). Measuring these latencies implicitly, by aggregating the processing times, or using hooks at the point of capture and rendering, do not provide the complete latency noticed by the user. This measurement ignores key aspects like: the time taken to acquire the image, the time required to display the results on the device, data transfer times between the device and the machines, etc. To accurately measure the entirety of the observed latency, an external latency measurement system is required. To measure latency from the user's point of view, outside observer techniques treat the system as a black box, and measure the delay between the response of the system to any stimuli, generally by using external visual or audio sensors (Steed, 2008). For measuring latency, the outside observer techniques require exclusive access to the system, and measure the latency for the system within the scope of the stimuli. These techniques are applied in sensor based virtual environments, where the performance of the tracking methods is not influenced by the size of the object, or the activities performed. In 3D Presence, the performance of the reconstruction algorithm is dependent on the granularity and scale of the captured object; So, directly applying currently available outside observer techniques will either provide latency measures for the reconstruction of the entire scene (scene latency), or only for the stimuli object (base latency).

In this chapter, we present an outside observer technique that can measure latency while the user is using the system. The Strobe method introduces a small object in the scene as shown in Figure 9.1, that is captured by an external high speed camera and is captured, reconstructed, and rendered by the 3D Presence algorithm. The use of a small clearly visible

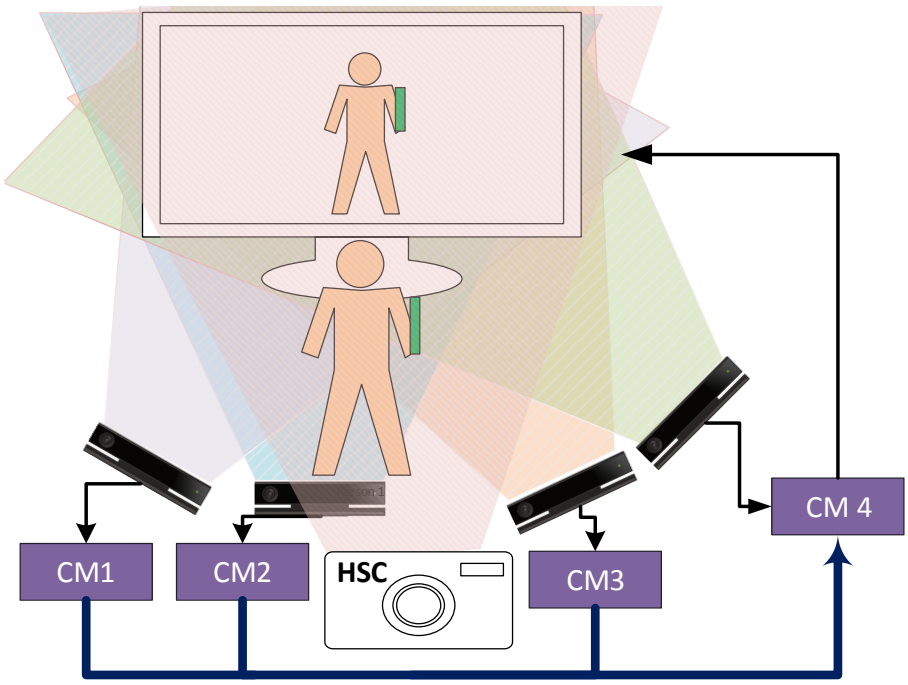

Figure 9.1. The Strobe approach setup to measure a multiple camera 3D presence system, using a green strobe and a high speed camera (HSC).

strobe, positioned away from the cameras, reduces the contribution of latency associated with the latency measuring method. The amount of overhead introduced by the latency measuring exercise depends on the exact location of the strobe in the scene. A high speed camera, capturing at $960 fps$, captures both the real world, and the virtually reconstructed, strobe in a single frame. The difference, between the real world and reconstructed strobes in the video, is used to determine the latency.

Evaluations of the method, on both the simulated and real world situations, highlight the fact that the Strobe method can measure latency to within a millisecond accuracy. The technique was applied to measure the latencies associated with: the raw color image capture from various RGB-D cameras, the rendering to different devices, a 3D Presence system that is reconstructing and rendering using 5 different approaches, etc. The observed latency, measured by the Strobe approach, was largely as expected, based on the measured implicit

latency. However, in certain situations, where there is lighting variation or usage of devices like projectors, the observed latency differed significantly from the implicit latency.

9.1 Related Works

Existing research, in the field of 3D Presence and 3DTI systems, focuses on the quality of experience or the quality of service that is measured internally over the network/internet (Kurillo and Bajcsy, 2013). All of these works either use the transmission delay, or combine all of the processing delays, to compute the system latency (Vasudevan et al., 2011; Raghuraman and Prabhakaran, 2015; Desai et al., 2015).

In virtual environments (VE), the level of user comfort is highly correlated with the latency of the system; So, internal techniques, like adding hooks to rendering API calls, are used to measure the latency (Chen et al., 2011). Steed attached a tracker to a pendulum, and tracked both the virtual object and the tracker using video cameras, to estimate latency (Steed, 2008). The motions of the objects are fitted to a sine wave, and the phase difference between the sine waves are used to estimate latency.

Modifying applications and adding hooks may not always be feasible, and in some instances, can add to the latency. To measure the observed latency of the system, outside observer techniques are used. The sinusoidal oscillations of a pendulum are used to measure the latency of a VE (Steed, 2008; Friston and Steed, 2014). A tracker is attached to the end of a pendulum and oscillated, resulting in the motion of a virtual object. These motions are captured using a normal camera (Steed, 2008), or a high speed camera (Friston and Steed, 2014); A computer vision program then measures the latency by fitting the motions to a sine wave, and estimating the face difference of the sine waves. Approaches that use high speed cameras simply rely on the number of frames between the motion of the real world and virtual objects, to measure the latency (Friston and Steed, 2014; Wu et al., 2013). Other approaches rely on external sensors, or use sound cards, to have higher frequency capture

rates, in order to measure latencies. Summaries of these methods can be found in (Steed, 2008).

Ohl et al. (Ohl et al., 2015) measured the observed latency of a 3D Presence system, by modifying the sinusoidal pendulum approach of (Steed, 2008). When they compared the observed latency with the expected latency, that was computed by adding up all of the delays, the expected latency was significantly less than the observed latency. They attributed these excess delays to the multiple monitor video wall, that was used for rendering the scene (Ohl et al., 2015). When captured by the cameras, the motion of the pendulum leads to motion blurred frames in the video, necessitating position approximations which reduces accuracy. The motion interpolation technology in many displays generate intermediate frames to smooth motions, making motion based latency estimations even less accurate, like (Steed, 2008; Friston and Steed, 2014). The Strobe method uses illumination changes with no motion, making it resistant to both motion interpolation and motion blur.

9.1.1 3D Presence

3D Presence systems capture the user using multiple cameras, reconstruct a 3D model, and render the model in the virtual world, all in real time for every frame captured by the cameras. A comprehensive list, of 3D Presence systems and approaches, is provided in (Kurillo and Bajcsy, 2013). For systems with RGB-D cameras, the data from each of the cameras is filtered to reduce noise, and then segmented to isolate the target object. The intrinsic and extrinsic calibrations of the RGB-D camera are used to project the depth image into 3D space, and find the corresponding pixels in the color image. Finally, a mesh is generated using a real time 3D reconstruction approach. The meshes are then realigned using extrinsic calibrations between RGB-D cameras, and rendered with overlap handled using a fragment shader, as described in (Vasudevan et al., 2011). There are many factors, like the number of cameras, camera fps, shape of the object, hardware speed, transfer rates, network speed,

etc., that effect the performance of the 3D Presence system. The main algorithmic aspect is the reconstruction technique used for generating the 3D mesh. So the latency associated with the following reconstruction and rendering approaches were estimated:

Bisection Meshing (BM) (Vasudevan et al., 2011): A low poly mesh is generated from the depth image, by bisecting triangles based on the standard deviation of the region inside the triangle. Bisection is stopped after reaching a minimum size. Whenever necessary, extra triangles are added to keep the mesh conformed.

Hessian sweep line Meshing (HM) (Desai et al., 2015): Meshing is performed using a sweep line image meshing approach on the percentage of points having high curvature values, as estimated by the Hessian operator on the depth image.

Image Meshing (IM) (Raghuraman and Prabhakaran, 2015): A dense mesh is generated by moving a square across the depth image, and adding a triangle if the corners of the square are close to each other, depth-wise .

Shader Meshing (SM): The IM approach is implemented at the geometric shader for efficient processing. The shader implementation eliminates the need for a merge operation when compared to a parallel implementation using CUDA; But shadowing and other multiple pass operations might result in re-meshing, which slows down the approach.

Unity Rendering (UR): The IM approach is applied on a scaled down (2:1) version of the depth image, to generate a low poly model that is rendered in Unity3D.

9.2 Strobe Approach

Outside observer approaches, used to measure latency, are system independent and rely only on the time between a stimuli, and the appropriate response from the system. In order to be able to capture both the stimuli and the corresponding response, these approaches require exclusive access to the system, whose latency needs to be measured. In the case of 3D Presence systems, the subject that is being reconstructed plays a vital role in the latency

of the system. So, directly measuring the latency of capturing the entire scene or just the stimulus object, does not provide an accurate estimate of the system latency while it is being used by the user. To measure the latency while the user is actively using the system, the Strobe approach uses a non-intrusive light emitting strobe.

The Strobe approach relies on the illumination change of the strobe to estimate system latency. By simply relying on the illumination change, the approach is not susceptible to capture angles, overexposure, display optimization, motion blur, noisy data, etc. The use of a high speed camera to capture both the stimulus and response, enables the Strobe approach to estimate latency with millisecond precision.

To measure the latency of a multi-camera system, the Strobe approach is set up as shown in Figure 9.1. The strobe object, emitting a signal at a constant frequency, is positioned away from the application cameras, so that the cameras have an unobstructed view of the strobe. The strobe is then captured, reconstructed, and rendered by the application. A high speed camera captures a video, containing both the real world and rendered strobe in a single frame.

The Strobe approach relies on the illumination change of the strobe, so the colored video is converted to grayscale for processing. Since a single video frame consists of an original and reconstructed strobe object, manual intervention is required to identify them individually in the frame. The region of interest, corresponding to the original and rendered strobe, are highlighted by the user and provided as input for the strobe signal identification. Since the strobe is located at a certain location in the scene, and the scene is rendered exactly the same way every time, it is required to identify the real world and virtual locations in the video only once per recording session. Each region then undergoes strobe signal identification and state recognition, resulting in a couple of rectangular waves; These rectangular waves are used to estimate latency.

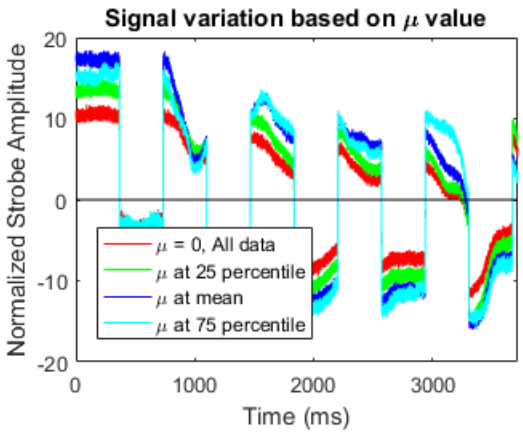

Figure 9.2. The effects of μ on the estimated strobe signal.

9.2.1 Strobe Signal Identification

To identify a strobe signal for a given region R, it needs to be ensured that all the pixels inside R correspond to the strobe. Expecting the user to mark the strobe region accurately, where all the pixels inside that region belong to the strobe, is extremely optimistic. However, it can be expected that the user is able to highlight the region that largely contains the strobe. Considering that the identified region largely consists of the strobe, a maximum likelihood estimate $S(p)$ can be estimated, where pixel p corresponds to the strobe. Since only the pixels corresponding to the strobe will change periodically in intensity, the variance of a pixel's intensity throughout the video can be used as a measure for $S(p)$.

The pixels with a $S(p)$ value greater than μ are identified with the strobe. The value for the strobe signal in each of the frames is estimated to be the average value of all of the pixels that contain the strobe. The value of μ is very important in situations where it is hard for the user to identify the strobe region accurately. While using $\mu = 0$ may not impact the accuracy of the state recognition in some cases, having an accurate μ enhances the signal significantly, as shown in Figure 9.2. After analyzing a large number of videos, it was found that the μ was set to the 75^{th} percentile value of the regions $S(p)$, and gave the best amplitude varying curves.

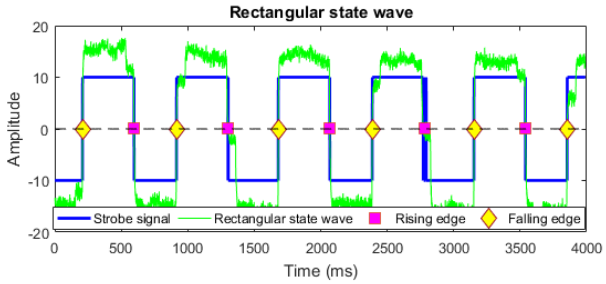

Figure 9.3. The effect of an incorrect σ on the rectangular state wave.

9.2.2 State Recognition

The strobe is either at a high state when it is on, or at a low state when it is off. It is known that the strobe is on or off for a fixed period of time, emitting a periodic signal that is captured. The identified strobe signal should ideally be a square wave. The signal might get distorted to a rectangular wave when combined with application latencies. Due to the various artifacts introduced by noisy capture, overexposure, bad calibration, incorrect 3D reconstruction, inaccurate rendering, etc., the wave might consist of smaller rectangles.

The periodic nature of the signal, even in situations with obtuse delays, can be used to transform the signal to a rectangular wave. This transformation can be performed by applying a threshold, σ, that divides the signal into highs and lows. Since the signal is supposed to be largely periodic, σ can be roughly approximated as the mean of the signal. In certain situations, like the signal shown in Figure 9.3, the value of σ may not produce a largely square signal. This can be easily identified by calculating the duty cycle of the rectangular wave.

If the detected lacerations are concentrated in a small region of the signal, then the edge is identified at the mean of the region. However, if the breakages are well distributed in the signal, then the first, or third, quartile is used as σ. If neither of these values is able to transform the signal into a largely periodic rectangular wave, then the signal is tagged invalid and ignored.

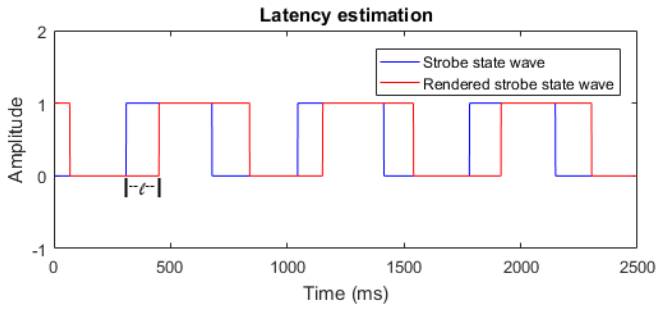

Figure 9.4. The local latency (l) that is measured between the state waves, W_R and W_D.

9.2.3 Latency Estimation

The latency is estimated as the time between the edges of the real world strobe state and the rendered strobe state. Figure 9.4 shows a couple of rectangular state waves, W_R for the real world strobe state wave, and W_D for the displayed strobe state wave. The local latency (l) is the temporal difference between the occurrence of a rising edge on W_R and the corresponding rising edge on W_D, or the temporal difference of the falling edges between the two waves. The average of all of the individual latencies, for the span of the video, is the observed latency of the system for that activity.

3D presence systems generally rely on RGB-D cameras that capture as an approx rate of $30fps$. When the strobe goes from the on to off state, there is a high likelihood that due to the low shutter speed of the RGB-D cameras, the frame capturing the transition is overexposed, as shown in Figure 9.5. This overexposure would result in the difference in the period of the high (positive) side, and the low side of the wave. In situations where a static scene is captured, it would be more appropriate to use the difference in the rising edges of W_R and W_D to estimate the latency, then using both the rising and falling edges together. Since the approach is meant to measure activity latency, other parameters are not constant during the measure of latency, so both the rising and falling edges are used for latency measurement.

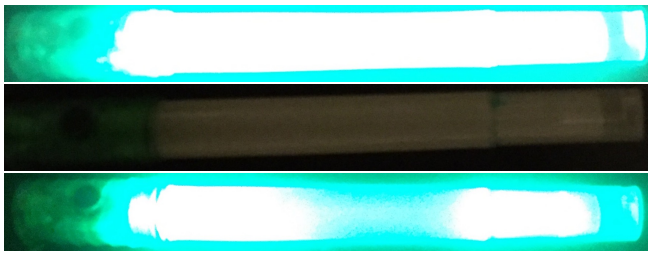

Figure 9.5. Various states of the strobe from top to bottom: Strobe turned On, turned Off, and the overexposed capture of On to Off.

9.3 Implementation

The 3D Presence system is setup using multiple camera machines (Intel i5 5600 @ 3.2 GHz, 16GB DDR4 RAM, NVIDIA GTX 1060, Windows 10) and a rendering machine (Intel i7 6600k @ 4.5 GHz, 32GB DDR4 RAM, NVIDIA GTX 980 TI, Windows 8.1). All of the machines are connected to an individual MS Kinect V2 sensor for data acquisition, from a calibrated set of sensors capturing a scene. The rendering machine renders on a $1ms$ response time $144Hz$ 24in Acer monitor, or a Samsung 3D LED TV. All of the camera machines have gigabit networking, with the rendering machine having a 10GbE connection. A Sony RX10 III is used to capture the high frame rate 1080p video at 960 fps.

The 3D Presence system code is written using C++ and CUDA, with rendering done in OpenGL 4.5, and shader written in GLSL. The Unity3D rendering is written in C# and connected to the rest of the framework using sockets. All rendering is performed at the V-Sync rate to avoid refresh related artifacts on the display. The latency estimation is done by processing the video in MATLAB, and the source code is available upon request. Network Time Protocol (NTP) is used to synchronize the clocks of all of the machines, to be millisecond precise.

9.4 Evaluation

The Strobe approach relies on the frequency of the strobe light, accurate state recognition, and the frame rate of the high speed camera, to measure the latency precisely. The accuracy of the Strobe approach and its vital aspects is validated as follows:

Camera Frame Rate: The high speed camera is configured to capture at a frame rate of $960fps$, resulting in an inter-frame interval (IFI) of $1.04ms$. Directly measuring IFI, using a $1KHz$ visual signal, may be inaccurate due to the likelihood of overexposure in each of the captured video frames; So, a $100Hz$ signal is used to measure the IFI. Multiple fixed duration videos, of an alternating black and white screen displayed on a $1ms$ response time $100Hz$ monitor, are captured using the high speed camera. It was verified that all the videos contained an equal number of frames. The frames, between the first and last state change in the video, were considered for estimating the IFI. The average number of frames between the black and white state change of the monitor was 9.59, with a standard deviation of 0.49, yielding a IFI of $1.046ms$, which is almost the same as the expected IFI of $1.042ms$.

State Recognition: The Strobe approach uses the change in illumination to estimate the state change. Since no color information is used, an alternating black and white screen is used to simulate the strobe signal. A monitor, with $1ms$ response time, was used to generate the strobe signal at varying frequencies of $25Hz, 30Hz, 50Hz, 60Hz$, and $72Hz$. Multiple videos, of varying frequencies of the strobe signal rendered on the monitor, were captured using the high speed camera. Despite the low response time of the monitor, it is likely that the video might contain frames that were captured when the monitor content was being refreshed. By using a large region size to estimate the strobe signal, the influence of monitor refresh and other display related artifacts are largely avoided. The time period of each of the states were identified to within $\pm 1ms$ of the expected value.

Simulated Delay: A reference signal with a period of $600ms$ was used to measure simulated latencies, between 1 and $299ms$. The Strobe approach was able to identify all of

the latencies accurately. On further evaluation, it was verified that the Strobe approach is able to identify latencies that are less than half the time period of the strobe signal. The curves for the simulated delay were generated directly by the system.

For real world validation, a monitor running at $100Hz$ was used. A signal time period of $600ms$ was displayed on a monitor with a response time of $1ms$, running at a frequency of $100Hz$. An alternating black and white region was shown on the display, which corresponded to the low and high value of the signal. Another signal, generated by delaying the original signal, was also displayed similarly on a different region on the same monitor. In order to verify that there was no delay between the rendering of the signals, both of the signals were rendered on the display with no delay. As expected, it was found that, as long as the monitor refresh interval is a multiple of half the time period of the signal, the signal is rendered and interpreted accurately by the Strobe approach. Due to this reason, all the delays tested were multiples of the refresh interval (10ms). The Strobe approach found the latency between the signals to be 9.65, 19.94, 49.8, 99.76, 149.64, 199.6, 249.44, and $289.45ms$ respectively, for a simulated delay of 10, 20, 50, 100, 150, 200, 250, and $290ms$ respectively.

Strobe Frequency: In all of the above evaluations, a monitor was used to generate the strobe signal. Depending on the 3D reconstruction approach, large objects, like the monitor, can take a significant amount of time to reconstruct compared to smaller objects. Any strobe source with a time period greater than double the expected latency is sufficient to measure the latency using the Strobe approach; So, a small strobe LED (Figure 9.5) is used generate the strobe signal. To ensure that the LED signal is periodic and has a fixed frequency, multiple videos of the LED signal were recorded by the high speed camera. The time period of the signal was found to be $736ms$, with a maximum variation of $\pm 2ms$.

9.5 Experiments

The Strobe approach for latency estimation was applied to numerous situations related to 3D Presence. The latencies associated with raw capture, rendering, single camera reconstruction, and the entire 3D Presence are presented in the following sections.

9.5.1 Camera Latency

The capture and rendering of a color image, from an RGB-D camera, is dependent both on the camera and the software used to acquire and show the image. Three widely used RGB-D cameras, MS Kinect V1, MS Kinect V2, and Creative Senz 3D, were combined with the example tool kit code provided for each of them to render color images. The latency associated with all 3 cameras for capturing and rendering a color image in a well-lit room, on a $144Hz$ monitor, was found to be in the range of 49 to $57ms$. Under low light conditions, the Kinect V1 performed exactly the same, with similar latencies. However, the Kinect V2 and the Creative Senz 3D increased their exposure rate, causing the latencies to increase to around $88ms$. The 3D Presence system uses Kinect V2s for the acquisition of RGB-D data; In order to ensure timely acquisition, the capture scene was well-lit.

9.5.2 Display Latency

The time taken to render an image on screen varies between different types of hardware that are used for rendering. The MS Kinect V2, with the example color image display program, was used to render the live color feed of the camera on-screen. A 1ms response time monitor was connected to the machine with a display port cable, and run at different refresh rates. It was found that the latency for rendering the color image varied more than the expected refresh rate variations. With the monitor refresh at 144Hz, 100Hz and 60Hz, the latency for rendering the camera feed was 54, 67, and $78ms$, respectively. Similarly, when the rendering

was performed on a 60in 1080p 60Hz 3D Samsung TV, and a 1080p 60Hz 3D Canon projector was connected to the machine with 15ft HDMI cables, the latency was higher than the $78ms$ seen on the monitor. The TV had a latency of $93ms$, and the projector averaged at $104ms$. The response time of both of the devices was supposed to be around $5ms$, but is not clearly rated. The added 10 to $15ms$ difference, between the projector and TV, could primarily be due to the cable or the display technology. Since the high speed camera needs to capture the displayed/projected result, the added time could also be due to projected surface, or the lower luminosity for the projection at the beginning of the refresh. It is clear that the exact hardware that is used to render has significant impact on the overall latency of the system, so the monitor at $144Hz$, with a $1ms$ response time was used for all of the 3D Presence experiments.

9.5.3 Single Camera

A 3D Presence system, using a single MS Kinect V2 camera connected to a rendering machine, was used to capture and reconstruct an entire scene, with a variety of objects in it. All of the reconstruction approaches that are listed in Section 9.1.1, were individually used to reconstruct the scene, and both the implicit and observed latencies were estimated. Figure 9.6 shows the variation in implicit latency over time for a static scene, reconstructed using all of the approaches mentioned in the above sections. There was no discernible difference between the processing times and the implicit latencies for all of the approaches, with and without, the strobe object in the scene. It is clear that the Strobe approach does not add any new overhead to the 3D Presence system; Hence, the measured latency is equal to the normal functioning 3D Presence system, and no greater. The average observed latency for each of the approaches, BM, HM, IM, SM, and UR, are 156.5, 343, 104.3, 81, and $122.4ms$, respectively; And are comparable to their average implicit latencies of 90, 275.5, 26.8, 9.9,and $60.2ms$, respectively.

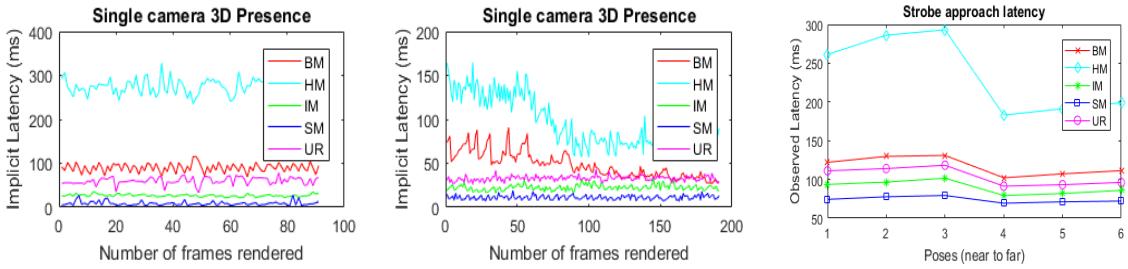

Figure 9.6. 3D reconstruction latency plots from left to right: Implicit latency measured for an entire scene, the implicit and observed latency for a person waving and moving backwards, away from the camera.

To analyze the variation in latency, when reconstructing just the user and not the entire scene, two different size users (small and large frames) were asked to use the system. The users were positioned at $1.25m$ and $2.5m$ away from the camera. At the closest distance, the pixels corresponding to the user in the depth image, were the largest. At the furthest distance, the pixels containing the user were significantly less. The users were asked to stand in 3 different poses at each location; The poses were: standing with their arms naturally at their side, the up arrow pose with their arms 45° to their body, and the T-pose with their arms straight out, at their shoulders. The users had to stay still and hold the pose, in order to ensure that the individual local latencies, extracted by the Strobe approach, are largely similar. Each of the poses resulted in a significant variation in the size of the bounding rectangle for the user, in the depth image.

The implicit latencies for the entire set of poses, at different locations, is shown in Figure 9.6. The latencies seem to vary significantly between locations, with minor variations for the poses at each location. This same trend is seen in the observed latencies for the same individual, as shown in Figure 9.6. The difference between the average implicit latency and the observed latency, estimated using the Strobe approach, remains fairly constant for all of the reconstruction and rendering approaches. The trend between the implicit and observed latency remains the same, for even the smallest framed individual. At the furthest distance, the latencies between the two users are quite similar, $\pm 5ms$. But at the distance of $1.5m$,

the latency associated with the larger person was as high as $10ms$ more than the latency of the smaller user. The differences in the latency, at the furthest distance ($2.5m$), were dependent on the amount of noise present in the depth image capture of the person. In certain situations, big chunks of the larger user were not getting detected by the camera, leading to lower latency.

9.5.4 Multiple Camera

Multiple cameras were positioned, one on top of the other, capturing the exact same scene as in the single camera case, mentioned above. For as many as four cameras, the latencies, both implicit and observed, for all of the approaches, increased by $30-35ms$ when compared to the corresponding latency in the single camera setup. This trend continued when the user was captured the same way as described in the previous section. The latency variations were again in the range of $20-30ms$. The transmission of data between machines, is performed by compressing the color image to JPEG. The compression and decompression process, along with the transmission time, are totally responsible for the excess delays noticed in the multi-camera case. There were no significant processing or rendering latencies added on, largely due to the efficient multi-threaded architecture of the 3D Presence system.

When the cameras are positioned to capture the 360° view of the user, the latency was equivalent to the maximum latency associated with the capture, reconstruction, and transmission of the camera closest to the user. This is similar to the situations represented in the single camera case, along with the added transmission latency.

CHAPTER 10

VISUAL QUALITY VS RESPONSIVENESS

Interactive 3D Tele-immersion (i3DTI) systems enable collaborative augmented virtuality, by allowing geographically distributed users to see and interact with each other in a virtual world. Each user is captured using multiple cameras to generate a 3D mesh of the user every frame. These meshes are then transmitted and rendered at different sites to allow communication between the users. Even for a single camera, large quantities of data (approximately 6MB) are captured, processed, and transmitted every frame, resulting in low frame rates and a perceivable lag even on high speed networks. So, even the current state-of-the-art 3D Tele-Immersion (3DTI) applicatons for gaming (Wu et al., 2010; Raghuraman et al., 2012; Venkatraman et al., 2013) are restricted to a single RGB-D camera, thereby limiting the possible in-game virtual camera and user interactions.

Multiple camera capture is vital to support user customizable viewing options in i3DTI applications. This compounds the latency of the application, prior research (Wu et al., 2010) recommends a maximum latency of 120ms for ensuring user engagement. To achieve such low latency, compromises in Visual Quality (VQ) need to be made. Mesh simplification can reduce the mesh size, but of is time consuming and unsuitable for real time applications. The fastest compression/decompression methods require more than 100ms per mesh, increasing latency significantly.

Our i3DTI framework uses GPU acceleration to speed up processing, and distributes the work load across multiple machines to reduce processing times drastically, and allow a 4 camera system to respond in under 50ms. Natural full body user interactions are realized by using just the skeleton of the user. A virtual camera view based rendering approach is used to select a small subset of cameras that need to be processed for rendering. Although this reduces the delay significantly, it is still not sufficient in situations where there are many cameras with overlapping views. By determining if users playing an engaging game require,

Figure 10.1. Third person view of the striker kicking the ball towards the goal in the penalty shootout game.

or even notice, the change in quality of their 3D reconstructed models, lower resolution meshes can be generated to reduce system latency significantly. Prior studies (Wu et al., 2011) have focused on just the VQ of the rendering in 3DTI, or the effects of latency on user engagement in a 3DTI game (Wu et al., 2010).

In this paper, a two player i3DTI penalty shootout game as shown in Figure 10.1 is developed to study the impact of low latency and high VQ on the user's experience. Two versions of the game, one with high VQ and the other having low latency, are evaluated by a user study. To eliminate the bias due to the other player, the single player version of

the games is used for all of the user study participants. The subjective evaluations of the users, along with the system performance and in-game player performance scores, provide interesting insights to answer the rendering quality question.

10.1 Related Work

Virtual Reality (VR) applications have been used to study or improve player performance in ball games. The major obstacle for wide scale adoption of such applications is the latency associated with the systems. A comprehensive survey of these efforts can be found in (Miles et al., 2012).

Approaches for real time 3D reconstruction using multiple cameras in 3DTI have been studied extensively and beyond the scope of this paper. A survey of recent 3DTI related reconstruction approaches and applications is given in (Kurillo and Bajcsy, 2013). The first 3DTI gaming application "I'm a Jedi" was presented by (Wu et al., 2010). This game allowed interaction using a light saber between remote players. A single stereo camera was used per player, and no virtual world objects or physics were used in the game. An immersive tennis game, using body sensors to control the racket, was shown by (Raghuraman et al., 2012). A Kinect camera was used to capture the player's back, and a point cloud was rendered using OpenGL. Basic collision detection was performed to allow the players to hit the ball. The baseball game developed by (Venkatraman et al., 2013) consisted of two Kinects, each capturing a different player. Virtual objects, like a baseball bat and ball, were placed in the players' hands, based on the joint information provided by the Kinect. Players used gestures to throw the ball and due to limited cameras, only a fixed view of the scene was provided. All the above games use a single camera per site to capture the player, reducing the processing time and overall complexity. The players are rendered inside the scene, but their interaction is restricted to certain joints, or are dependent on external devices like

sensors in (Raghuraman et al., 2012), or the light saber in (Wu et al., 2010). Our game allows multiple angle rendering and whole body interaction.

The effect of the VQ of the mesh on user experience was studied by (Wu et al., 2011). The study focused on how much deterioration is inconceivable by the user, allowing for lower quality meshes to be transmitted without any change in user perception. Rather than letting the user focus only on the rendered information, we engage the users in an interactive activity and study their response on various questions related to their overall experience, including VQ.

10.2 i3DTI Framework

A i3DTI system allows geographically distributed users to be present in and interact with the virtual world, using their body. To achieve 3D presence, the user is captured from all directions using multiple calibrated RGB-D cameras simultaneously, to generate a 3D mesh that can be virtually rendered. The i3DTI framework is designed to address the major needs of any i3DTI system, and allow rapid i3DTI application development. The i3DTI framework handles not only the capture, reconstruction, transmission, and rendering of the user, but also enables the natural full body user interaction with the system. For each of the cameras, the framework uses a rendering pipeline and an interaction pipeline, as shown in Figure 10.2.

10.2.1 Rendering

The depth image returned by the RGB-D cameras is noisy, and the characteristics of the noise vary from camera to camera. Depending on the noise, a median/bilateral or similar filter is applied on the depth image, to improve image quality. The user can be isolated in the depth image by using region based segmentation, or similar approaches. The depth image is converted into a 3D point cloud, using the intrinsic parameters of the depth camera. The point cloud, along with the neighborhood information from the depth image, are used to

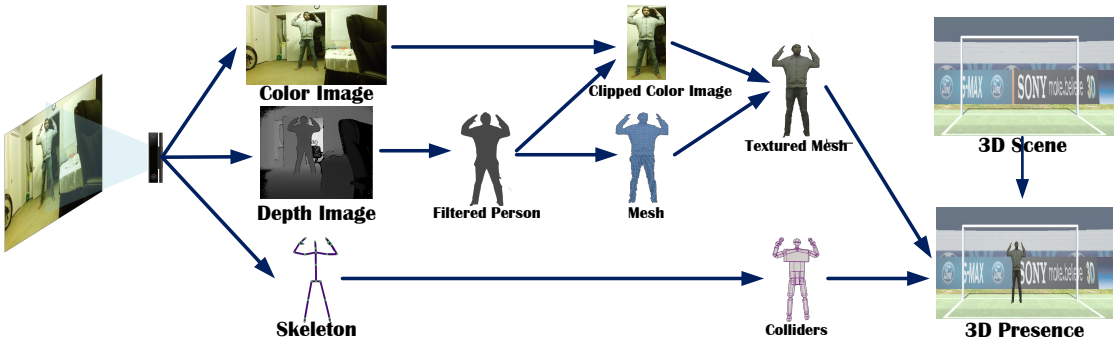

Figure 10.2. Reconstruction pipeline for a 3D immersive application using MS Kinect V2 for capture, and Unity3D for rendering.

generate the 3D mesh by applying a very fast image based meshing approach (Raghuraman and Prabhakaran, 2015). The color image is then mapped to the vertices of the mesh, using the extrinsic calibration between the color and depth cameras to create a fully textured 3D mesh. The color image is cropped to the relevant size, and compressed to JPEG on the GPU, to reduce the data size for faster transmission.

Even while rendering on 3D displays, depending on the RGB-D camera arrangement and the position of the virtual camera, only a small subset of meshes generated from certain cameras are required to be rendered. A view based camera selection approach (Pulli et al., 1997) is provided by the framework to select the meshes to be rendered, based on the virtual camera view. For faster performance, only the camera machines providing the selected meshes need to capture, process, and transmit the data. For example, view based rendering achieves better quality rendering, as shown in Figure 10.3, by adding only 25% (8 ms) to the latency, in comparison to rendering only the front and back meshes. For a given camera setup, the rendering quality can be varied by changing the level of details, and the degree of overlap required for rendering a mesh. Rendering overlapping meshes directly leads to many rendering artifacts. A fragment shader similar to (Vasudevan et al., 2011), that considers the camera capture, virtual camera, surface normal, and virtual lighting to estimate the color to be rendered, is used to eliminate the artifacts and blend the model inside the scene.

Figure 10.3. Side view of the player mesh using view based rendering (left), and front and back mesh rendering (right).

The model's interaction with virtual lighting produces shadows, improving user's depth perception.

10.2.2 Interaction

The framework provides full body interaction using physics colliders. In an i3DTI system a new mesh is created every frame, and estimating mesh based colliders, every frame, will add significant latency. So instead of waiting for the mesh, the skeleton of the user is used to create the colliders. The skeleton is either provided by the camera (MS Kinect V1 and V2), or can be estimated from the depth image. The accuracy of the detected skeleton is susceptible to noise and occlusion. By orienting the camera to see the user completely, the detected skeleton accuracy can be increased. Even though the speed and orientations of the joints, and the corresponding collisions may not be accurate all the time, our observations show that this model is still reliable. Any i3DTI environment requires the person to see the screen, thereby forcing them to face the front camera at all the times; This reduces the possibility of severe occlusion, which results in good skeleton detection.

Fast collision detection is achieved by covering the body with a combination of box and capsule colliders. Box colliders map the chest and abdomen region of the person. All

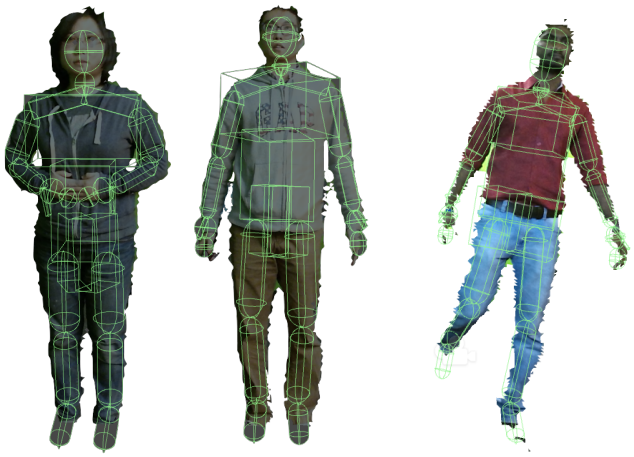

Figure 10.4. Skeleton based colliders are shown in green for players of different size.

the other parts, including the head, are represented by a capsule collider. Since the skeleton represents the medial of the person's mesh, it is possible to extract a reasonably good estimate for the size of various parts of the body using just the skeleton. The human body is largely symmetrical, and many parts of the body are proportional to each other. For any given skeleton, a joint association based model is used to estimate the size and orientation of each collider. The entire human model is made collision capable in microseconds, without needing any mesh or point cloud information, as shown in Figure 10.4.

10.3 The Game

The combination of the rendering and interaction pipeline of the i3DTI framework makes it possible to create very engaging games with good quality graphics. To study the preferences of the user, in terms of visual quality and the level of interaction, the game needs to cater specifically to highlight the benefits and minimize the limitations of the i3DTI technology.

10.3.1 Game Requirements

Like video games, an i3DTI game needs to be intuitive, easy to play, engaging, etc. The fundamental benefit of an i3DTI game is the ability to embed the user directly into the game.

Figure 10.5. Play area setup, with multiple MS Kinect V2 cameras (marked in red) for capture, TV, and projector for rendering.

This results in a one-to-one correspondence between the user's expressions, movements, actions, etc., in the real and virtual world. The capture area in the real world is limited in size, as shown in Figure 10.5; To keep the game realistic, and maintain the direct correspondence, the motion of the player in the game should also be limited. Occlusion, while capturing the user, results in both an inaccurate skeleton and a reconstruction with holes and other artifacts. To reduce the amount of self occlusion, the actions required to play the game should position the user in poses that can be captured clearly by all the cameras. Using external input devices causes visual occlusion, and provides a less engaging experience.

10.3.2 Penalty Shootout

Many different games can satisfy the requirements listed in Section 10.3.1. The penalty shootout situation in soccer meets all of the criteria, and also allows for individual practice play. The penalty game is played by two players, positioned as the striker or the goalkeeper. The striker is supposed to send the ball through the goal post by kicking or using any other

part of the body, except their hands. The goalkeeper is expected to stop the ball from going into the net, using their entire body. The striker is positioned at the penalty spot and the goalkeeper is on the goal line, as shown in Figure 10.1.

The player is provided a default third person point of view at the start of the game, and can change the virtual cameras orientation and location to match their preference. To improve the game play, the player is rendered translucently, allowing them to see the ball and other objects through their body. The texture of the player model is altered based on the virtual scene lighting, to provide realistic rendering. An accurate shadow of the player is rendered by positioning multiple virtual light sources. The shadows in the scene enhance the user's capability to track the ball and other objects in the game.

10.3.3 Mini Games

While penalty shootout can be played and enjoyed without prior knowledge or skill, possessing good skills enhances the two player gaming experience. The mini games allow the players to practice alone, to improve the necessary skills needed to play the penalty shootout game. The mini games were equally challenging and required slightly different skills. They also allowed for an independent assessment of the player's perception about the system, and the skill level of the player. The following three mini games were created, with increasing level of difficulty:

Goalkeeper: The player learns the basic skills necessary to be a goalkeeper, like moving in the virtual world, ball tracking, and blocking the ball with the body. The ball is placed at the penalty spot and the ball's projected target region is shown to the player a second before the ball is kicked, as shown in Figure 10.6.

Targeted shooting: This game is designed to increase the precision of basic skills that were developed in the previous goalkeeper mini game. The player is supposed to kick a static ball, positioned at the penalty spot, as shown in Figure 10.6, into the net to hit the targets.

Figure 10.6. From left to right: Goalkeeper, with the circle showing the expected ball position, and the striker with practice targets.

Moving ball shooting: The ball is placed in the field of view of the player and passed towards them. The player is expected to either stop and kick, or directly deflect the ball, to hit the targets inside the goal. To be successful the player needs to be able to track the ball, move appropriately, time and direct the kick accurately.

The inherent system lag, 3D reconstruction quality, collision accuracy, real to virtual world correspondence, rendering quality, etc. are important factors effecting in-game player performance. Playing the mini games forces the players to be engaged and adapt to the inherent deficiencies of the system; This helps to ensure the player's ability to evaluate the system confidently.

10.3.4 Implementation

The i3DTI setup consisted of a total of 12 MS Kinect V2 cameras that were positioned around the room, as shown in Figure 10.5. Each of the cameras are connected to a camera machine (Intel Xeon 3.0 ghz processor, 18GB RAM and Nvidia Quadro 4000 graphics). For the two player penalty game, the capture area was divided equally into two sites, each containing 6 cameras to capture the user. At each site, 3 cameras were placed in front of and

behind the user, so that each side of the user was captured by 2 cameras. The cameras are pointed downward to capture the person from head to toe. Each site contained a rendering machine (Intel i7 2.4ghz processor, 32GB RAM and Nvidia GTX 970 graphics) that was connected to either a 3D TV or projector at each site. All of the machines are connected to a local switch, using gigabit networking for the camera machines, and 10GbE networking for the rendering machines.

The i3DTI system is implemented using massively parallel architectures in C++ and CUDA, to minimize system latency. The work load is also distributed across a cluster of machines used for both capture and rendering. The camera machines reduce noise using a median filter, segment the user and generate a 3D mesh from the depth image. The camera machines also generate texture mappings between the color image and the 3D user mesh, crop the relevant regions, and compress the image using JPEG. The penalty shootout and related mini games were developed in C# on the Unity3D gaming engine. TCP Sockets were used to communicate between the i3DTI framework and the game. The MS Kinect positioned in front of the display was used to track the skeleton of the user. The game was rendered to the user in stereoscopic 3D, on either the TV or projector, depending on the site. The system clocks of all the machines in the i3DTI setup are synchronized using NTP, to within a millisecond. The system latency is measured as the time between camera capture and the corresponding mesh rendering.

10.4 User Study

To answer the question of whether users prefer higher VQ or better interaction, a multiple stage user study was performed using the penalty shootout game. A total of 39 unpaid volunteers (21 male, 18 female), with varying backgrounds in the age group of 20 to 25, were recruited to be part of the user study. All user study participants were given an initial survey to determine their experience levels. Based on their replies, the study group consisted of 26

computer gamers, 15 regular soccer players, and 20 VR users. Only 9 of the participants indicated that they had neither experienced or seen 3D presence in action.

The overall user experience for a player in a two player game is too dependent on the other player. Similar to real world soccer, penalty shootout favors the striker, putting the user playing as the goalkeeper at a disadvantage; As a result, the user's opinions about the system, based purely on the experience of playing the two player penalty shootout game, may be biased. So instead of making all of the participants play each other in the two player game, and evaluate all the aspects of that experience, the single player mini games are used for detailed questioning. The user study consisted of a mandatory section involving single player mini games designed to answer the higher VQ or better interaction question, and an optional section involving the two player penalty shootout game, that was played by 31 participants.

10.4.1 Mini Games Study

Two versions of the mini games were created 1) A view based rendering version for Optimized Visual Quality (OVQ) and 2) A low latency version for Optimized Interaction Quality (OIQ). The participants first finished all of the OVQ mini games, followed by all of the OIQ mini games. The player's performance and system latency were monitored during each of the games played. A holistic view of each game played is provided by the subjective user evaluations, quantitative player's performance, and qualitative system latency information. All of the user study participants played the games in the order described in Section 10.3.3. Each participant was allowed to play a minimum of 10 turns, or more if requested by the user. Players were allowed to customize the virtual camera view using verbal instructions while playing the game. After playing each mini game, the user rated various aspects of their game play experience on a 7 point Likert scale, ranging from strongly agree (3) to strongly disagree (-3).

Optimized Visual Quality

The OVQ version of the games is configured to use view based rendering for displaying the user. The view based rendering of the user is captured by 6 uniformly distributed cameras, results in, at most, 3 camera meshes being rendered for most viewing angles. The user study questions were focused on the user's perception of their own performance and system aspects. Many of the players with no experience in gaming, VR, soccer, or 3DTI showed poor technique while playing the games. These players struggled to kick a goal, or even stop a ball heading towards the goal. The user ratings, grouped by question category, are shown in Figure 10.7. Based on the user ratings, it is clear that the users are happy with the performance of the system. The lowest user ratings were given to the questions that related to ball interactions; Considering the user group had only a few soccer players, others seem to require a greater amount of time to become acclimated to the system. After each game, the users were asked to rate the responsiveness of the system. In many of the cases, the user responses for the 3 questions varied by as much as 5 points; These did not correlate with the system latencies measured during the same game play. This clearly shows that a few users, with no VR or gaming experience, are not able of quantifying the responsiveness of the system.

Optimized Interaction Quality

The OIQ version of the games uses only the meshes from the front and back cameras to render the user. When the player model is viewed from any angle except straight from the front or back, a large hole in the side is visible, as shown in Figure 10.3. The same set of questions as Section 10.4.1 were answered by the user after playing the games in OIQ mode. The inexperienced players continued to struggle to perform even the simplest of tasks, like kicking the stationary ball. The difference between the average system latencies, of OVQ and OIQ versions of the games, were only 8ms. Compared to the ratings of the OVQ version, the

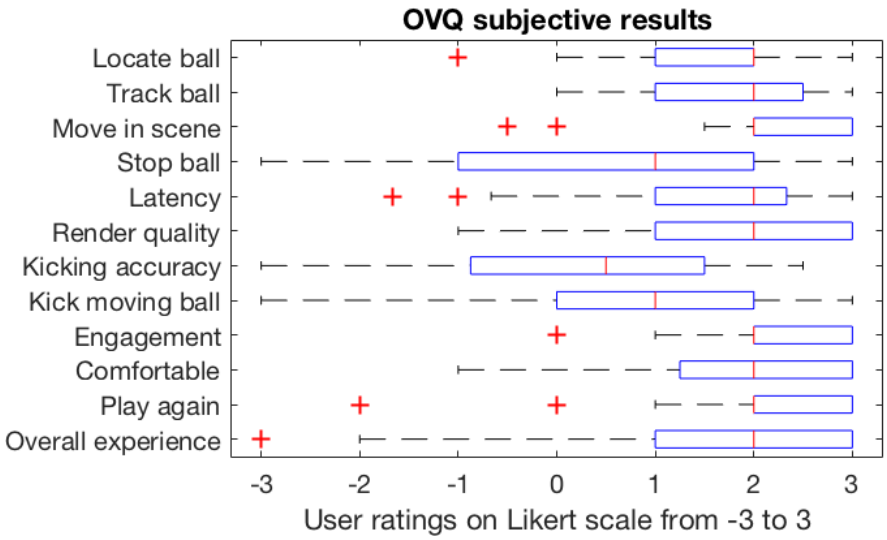

Figure 10.7. Box plot of user ratings for OVQ study

OIQ version's user ratings, showed in Figure 10.8, displayed a clear improvement in all of the categories, except tracking the ball. The OIQ game versions use only two cameras, leading to visible holes in the side of the body; Still there was an overall increase of about 0.5 points for the user rendering quality. Considering there were no changes in physics or interactions between the OVQ and OIQ game implementations, the improved user ratings for the ball interaction questions can be attributed to the user's ability to adapt to the system. Based purely on the user perception captured in the user study, it is clear that the users prefer faster interactions over VQ while playing an engaging game.

If only the subjective views of the users are considered, then most of them were able to play the game very well. However, the quantitative results of player performance provide a totally different outlook. There was no significant difference in the overall player performance between OVQ and OIQ implementations, as seen in Figure 10.9. The perception of improvement in kicking accuracy, shown by the subjective user ratings, turn out to be baseless. There is no evidence of an increase in kicking accuracy between the OVQ and OIQ implementations. This shows that the players engaged in playing the game seem to lose a sense of perception in the real world. In situations where the user played longer, the

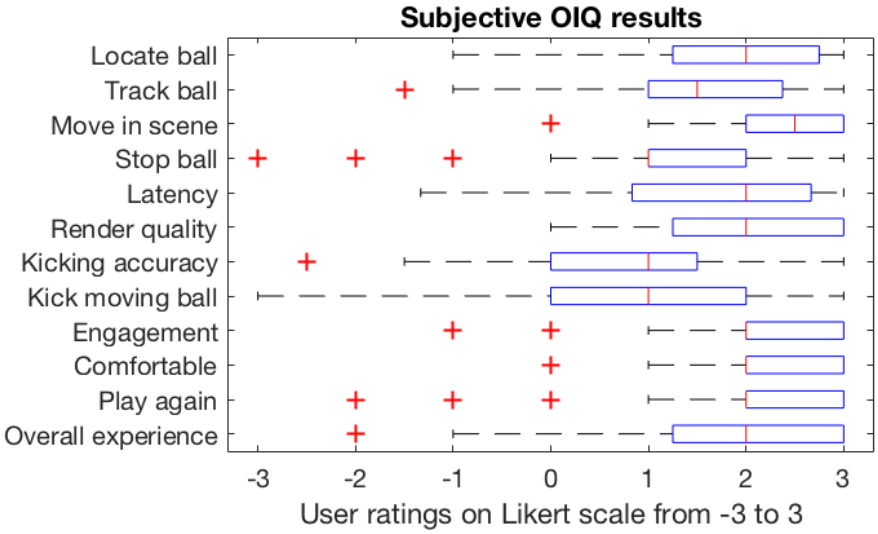

Figure 10.8. Box plot of user ratings for OIQ study

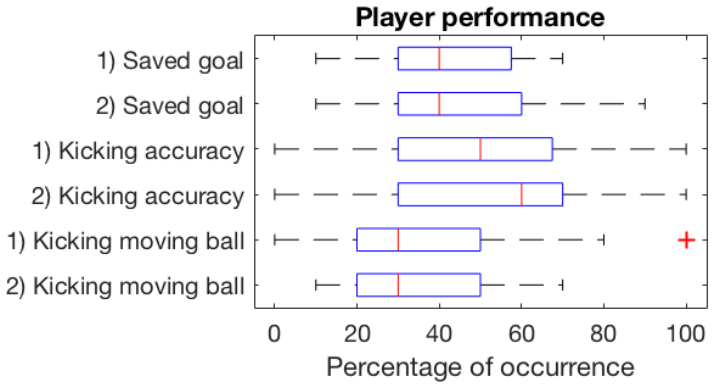

Figure 10.9. Quantitative statistics of player performance for 1) OVQ and 2) OIQ.

player performance showed significant improvement, between the OVQ game play and the corresponding OIQ game play. A few users with no prior gaming, VR, or soccer experience seemed to struggle even to kick the ball. Players having difficultly locating the static ball, struggled in both situations and rated the system poorly. The skeleton tracking of the foot is dependent on the footwear, and kicks made by users with shiny shoes were going in unintended directions. Overall, despite the low user ratings given for the kicking accuracy, some users with soccer experience were able to kick the ball accurately, while some with no gaming or soccer experience struggled.

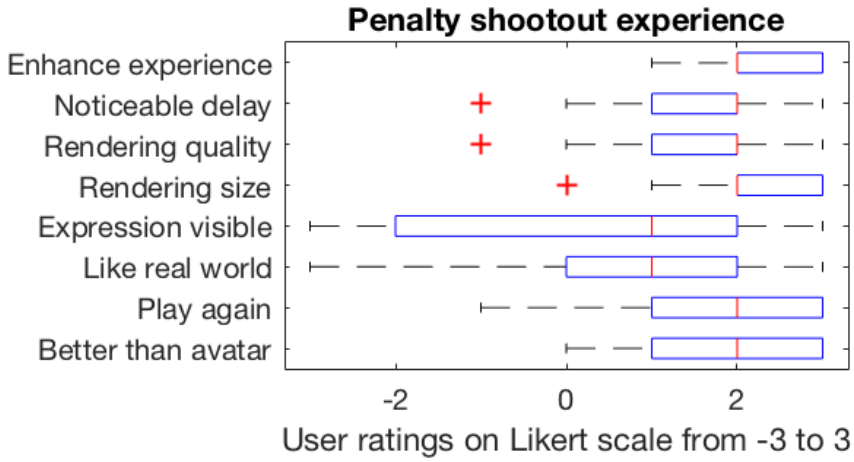

Figure 10.10. Box plot of user ratings for the two player penalty shootout game.

The difference between the subjective user ratings, quantitative player performance, and the system latencies clearly indicates that the user is completely immersed in the game. With the focus primarily on improving their skills and better game play, the player loses a sense of time and develops an elevated sense of performance. This trend can be clearly seen by comparing the results of the OVQ and OIQ user studies, especially in the case of mesh rendering quality, where a mesh with holes on the side is rated higher than a mesh without it.

10.4.2 Penalty Shootout Study

The penalty shootout game was setup using two adjacent sites connected by a high speed network. The game was implemented to be OVQ, and based on the camera setup, rendered 6 captured meshes at a time. The penalty shootout study pitted two randomly selected players against each other. The first player was given 10 turns to kick the ball, while the other defended the goal; The players then switched positions and repeated the task. The user study questions focused on the VQ of the other player's model, the perceived latency, and their overall quality of experience. All of the users agreed that there was little to no perceived latency, as shown in Figure 10.10. The study of the latency logs found that the

average latency for the local site was 43 ms, and was 67 ms for the remote player. There were some jitters, up to 30 ms, that were noticed during a couple of the sessions, corresponding to the lower user rating. All of the users strongly agreed that the presence of the other player in the game enhanced the user experience. Almost all of the users felt immersed in the game, and preferred this game over a similar game with virtual models. Despite the lopsidedness of the penalty shootout game in favor of the striker, users enjoyed it enough to want to play again. Players with no soccer experience felt the game was like playing real soccer, but the soccer players strongly disagreed. While everyone agreed that the rendered model was of good quality and properly sized, most of them were unable to see the expressions on the other player's face due to the distant virtual camera positions.

PART IV

APPLICATIONS

All of the research presented so far has real world implications. Expecting someone to implement each of the approaches, in a case by case basis depending on the kind of system they want to make, adds unnecessary barriers and slows down development. The i3DTI framework combines all of the research elements involved with i3DTI into a modular and easy to use package. The framework abstracts the devices, network, and other complexities of a i3DTI system, thereby allowing for the development of new approaches without dealing with all of these specificities. The framework decouples the gaming engine and the bulk of the distributed processing, enabling even graphic designers to build i3DTI applications. We present some of the real world applications for tele-medicine, gaming, education, etc., that were created using this framework.

Chapter 11: Using the i3DTI framework, we created an application for the remote diagnosis of patients with upper body injuries, specifically those involving the shoulder and elbow regions. This application was deployed between UT Dallas and the Dallas Veterans Affairs Hospital. The application was used by a doctor to diagnose patients, and was found to be very accurate.

Chapter 12: The framework was used to create lots of different i3DTI applications. Some of the more interesting applications are briefly described in this chapter.

CHAPTER 11

3D TACTILE TELE-IMMERSION FOR TELE-MEDICINE

According to the Centers for Disease Control (CDC), there are nearly 34.86 million Americans living with limited upper body abilities, that affect daily activity (of Health and Services, 2015). Many of these affected people live in rural areas with no access to treatment centers. Tele-medicine applications would be able to provide better accessibility to healthcare for these patients. Current state-of-the-art tele-health applications for rehabilitation only allow audiovisual communication (Chan et al., 2016). Some applications go beyond traditional video conferencing, by using 3D Tele-Immersion (3DTI) (Kurillo et al., 2016).

3DTI enables collaborative augmented virtuality, allowing geographically distributed users to interact with each other in a virtual world, using their "live" 3D reconstructed virtual models. This extra 3D information allows the doctor to visualize and diagnose the patients better (Kurillo et al., 2016). For physical rehabilitation, the doctor has to both see and feel the patient's limbs. In the absence of physical feedback, the doctor has to rely on clinicians to physically evaluate the patient, and convey their opinion to the doctor. For accurate diagnosis, the doctor and clinician need to have a good rapport and understanding of each other's techniques, which is not always possible; this leads to the lack of availability of quality healthcare for remote individuals.

Using a haptic device, the patient can feel the doctor's actions and the doctor can feel the patient's reaction. Achieving this kind of seamless interaction between the patient and doctor is extremely challenging over the internet (Venkatraman et al., 2014). Some tele-operation approaches have been proposed (Maciejasz et al., 2014) that work over the internet, but these allow the doctor to control a robot and get the force feedback based on the robot's motions.

In this paper, we introduce a 3D Tactile Tele-Immersion (3DTTI) framework that enables bi-directional force feedback and motion, allowing scenarios such as doctor/patient

Figure 11.1. The rendering from the patient side, captured during an RPDS session.

interaction possible. The framework models the bi-directional haptic force feedback and motion problem, as two independent bilateral transparency problems. This allows both the haptic devices to allow motion and provide force feedback, based on the other side's motion, even over the internet in situations with less than 40 ms of latency. 3DTTI also provides audio and 3DTI capabilities for effective communication over the internet. Due to the large volumes of data processed and transmitted over the internet, 3DTI systems have low frame rates and significant delays. 3DTTI implements efficient algorithms in parallel, utilizing the GPU, and distributing the load on each side to efficiently process, compress, decompress, transmit, and render 3DTI data in real time at high frame rates.

The high frame rate, low latency performance of 3DTTI allows for the creation of tele-heath applications. Remote Physical Diagnostic System (RPDS) was developed using 3DTTI, to allow doctors to diagnose patients having problems with their upper limbs, specifically the shoulder and elbow regions. The system was implemented with the doctor and patient located in different cities, connected over the internet, and rendered together in the virtual world, as shown in Figure 11.1. The doctor was able to adapt the procedure used for in-person diagnosis, for diagnosing the patient using RPDS.

A preliminary study was performed using five healthy people, to refine the procedures used by the doctor, and to update the system to ensure better quality and comfort for all future users. To study the real world impact of the system, 15 patients, with shoulder ailments, were recruited, to be diagnosed with RPDS. During the trial, the patients were diagnosed in-person by a doctor, and then diagnosed using RPDS by another doctor. All of the users had a favorable opinion about the system. A high degree of correlation was found, when the in-person and RPDS diagnosis of the patients were analyzed. The high subjective user ratings and accurate diagnoses, highlight the usefulness and promise of RPDS.

11.1 3DTTI Framework

The 3D Tactile Tele-Immersion (3DTTI) framework allows highly interactive communication between geographically distributed users, by immersing them in a 3D virtual world, and enabling them to feel each other's movements and force feedback by using the haptic device. 3DTTI provides features for both seeing the users in a virtual world using 3D tele-presence, and also feeling the user's action, using haptic transparency.

11.1.1 3D Presence

The 3DTTI framework displays the "live" 3D model of the user in the virtual world, at high frame rates, by capturing the user using multiple calibrated RGB-D cameras. Creating a single mesh for the user at a site, requires central acquisition and processing of all of the data that is captured by each of the cameras, every frame. This centralized processing creates a performance bottleneck for the system. So instead of creating a single mesh for each site, a mesh is generated from each camera view. The images of the person, captured by each of the cameras, is reconstructed independently of each other, using the steps shown in Figure 11.2. The user feels the forces via the haptic device, so it is necessary for the haptic device to be present in the capture area, next to the user. The presence of the haptic device occludes

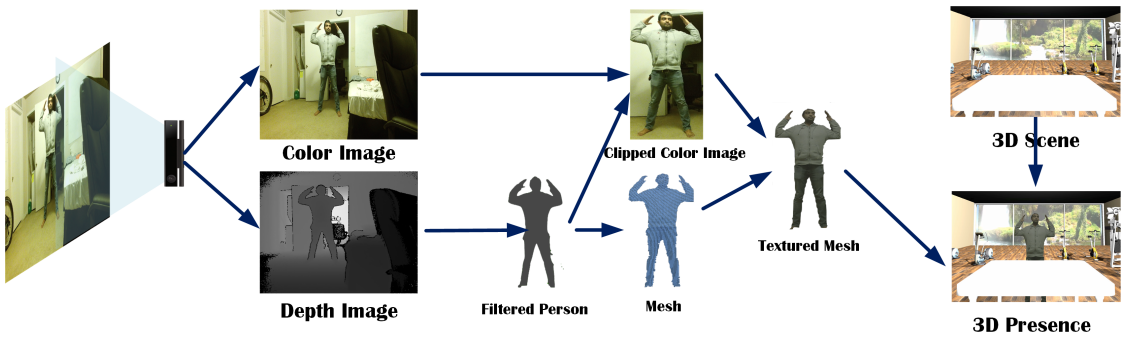

Figure 11.2. The steps involved in the 3D reconstruction of a person, captured using a MS Kinect V2 camera.

many parts of the user, making user identification difficult. The haptic devices need to be statically placed at fixed locations near the user, in the capture area. So the point cloud that is generated from the captured depth image, can be segmented volumetrically, to extract the regions of interest. The point cloud, corresponding to the regions of interest, is then surface reconstructed to produce a mesh. This mesh is transmitted and rendered in the virtual world, at the various sites. Since multiple cameras capture the same scene and generate different meshes, these meshes can overlap, leading to poor rendering quality if rendered directly. So a fragment shader that considers the capture camera view, virtual camera view, and the vertex normal, is used to handle the overlap and render the final result.

11.1.2 Haptic Transparency

In the case of remote diagnosis, and other contact based professions, it is important for the participants to feel each others hands. For creating such realistic touch feedback, the haptic devices are setup to ensure bidirectional haptic transparency. The bidirectional haptic transparency can provide the feeling of contact between the users, by replicating the forces applied on each side. In the 3DTTI framework, the bidirectional haptic transparency is setup, by estimating the force at each of the sites, based on the position of the local and remote haptic device. For a two site 3DTTI application, with a remote doctor diagnosing a patient, the haptic transparency can be established, as shown in Figure 11.3.

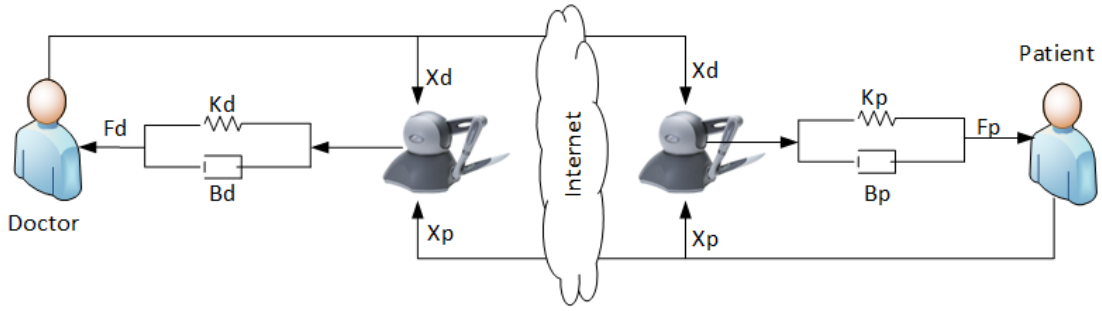

Figure 11.3. The haptic force feeback rendering for a two site transparency situation.

The force equation of both the doctor and the patient's site, are exactly the same. The force is estimated at the patient side as:

$$\mathbf{F}_p = K_p(\mathbf{x}_p - \mathbf{x}_d) + B_p\mathbf{v}_p \tag{11.1}$$

where $\mathbf{F}_p$ is the force feedback at patient's side, and $\mathbf{x}_p$ and $\mathbf{x}_d$ are the positions of the HIP, for the patient's side and the doctor's side. $\mathbf{v}_d$ is the velocity of the HIP, K_p and B_p is the stiffness parameter, and the damping parameter, respectively.

The haptic device is capable of producing some strong forces, that can lead to injury if the user is not careful. In the case of haptic transparency, the forces are generated based on the position of the remote haptic device; due to various reasons, like network delay, packet loss, etc., it is possible for undesirable forces to be estimated and applied. The bidirectional haptic transparency approach relies on the position consensus between the remote haptic devices in order to keep the forces small; if one of the users let go of the haptic device, then the device is completely controlled by the remote user, without any restrictions. This situation is highly likely in all of the applications, and special care needs to be taken to avoid user injury. To ensure the user is not harmed in any situation, a linear velocity based force cut off mechanism is used to halt the forces applied by the haptic device. Whenever the linear velocity of the haptic device, in the direction of the computed force, exceeds a preset threshold, the situation is deemed dangerous, and the forces are disabled.

The high frequency small size haptic data is typically transmitted using UDP. Due to their high frequency, any packets that are delayed, or lost, are simply ignored. Since certain networks do not allow UDP traffic, sometimes TCP needs to be used for transmitting the haptic data. When the haptic data is transmitted over TCP, the loss or delay of a single haptic data packet, can cause the entire haptic stream to be delayed; this causes longer transmission times for future haptic packets. While our haptic transparency approach can handle such occasional jitters, continuous long delays can lead to incorrect force estimations, making the haptic device dangerous to the users; this will trigger the force cutoff mechanism, thereby prompting a total tactile system shutdown. To avoid such a large pile up in the TCP stream while transmitting high frequency haptic data packets, instead of directly transmitting the haptic data that is acquired from the haptic device, an aggregation based haptic packet generation scheme is used to create haptic data packets. Whenever a pile-up is noticed during data transmission, the haptic data that is read from the device is averaged out over a period of time, to create a single packet. These aggregated packets are then transmitted and processed by the receiver, like any other haptic packet. The reduction in the frequency of the packets allows the system to quickly recover from large sporadic jitters, without causing significant haptic vibrations, or a complete tactile shutdown.

11.2 Remote Physical Diagnosis System

The Remote Physical Diagnosis System (RPDS) allows the doctor to physically evaluate a patient from a remote location. The doctor and patient are able to see each other in 3D, inside a virtual world, that allows them to move and see the world, from any point of view. Audio communication allows natural dialog between them. The haptic transparency of the 3DTTI framework enables the doctor and patient to feel each other's actions, using the haptic device. The combination of audio 3D visualization and haptic feedback, allows for seamless doctor/patient interaction, while enhancing the possibility of accurately diagnosing

the patient. While the system is capable of handling any physical diagnosis, the haptic device is designed to be used by the hand, so only upper body evaluations are possible. So, in this paper, issues specific to the shoulder and elbow are studied.

The doctor diagnoses the patient with upper limb problems, by evaluating their upper limb's range of motion and maximum isometric strength. The doctor studies the reaction of the patient, while performing the following ten basic motions: Elbow flexion and extension, arm elevation and depression, internal and external shoulder rotation, shoulder abduction and adduction, and shoulder protraction and retraction. The exact technique used to perform these actions can vary slightly from doctor to doctor. To evaluate maximum isometric strength, while the patient is performing the motions described above, the doctor applies resisting force to see how much force the patient can overcome. The doctor diagnoses the patient's condition, after taking into account the results of the patient's response to the resisting force, their range of motion, and the level of pain they experienced while performing the ten different motions.

11.2.1 In Person Diagnosis

During a normal patient consultation, the doctor asks the patient to perform ten basic motions, as shown in Figure 11.4, and are described as follows:

Elbow Flexion/Extension: The patient is asked to bend their arm at the elbow (flexion), and then straighten their arm (extension).

Arm Elevation/Depression: The patient stretches their arm out straight in front of their body, elevates to shoulder level, and is asked to lift their arm upward (elevation), and then to drop it downward (depression).

Shoulder Internal/External Rotation: While holding their elbow beside their abdomen, the patient bends their arm at the elbow to a 90° position; They are asked to move their wrist towards the body (internal rotation), and then move their wrist away from the body(external rotation).

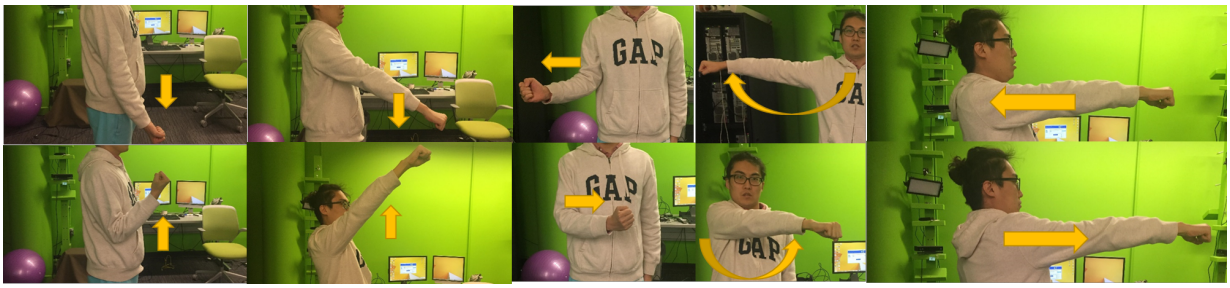

Figure 11.4. Arm motions for diagnosis shown top to bottom, left to right: elbow flexion, elbow extension, arm elevation, arm depression, shoulder internal rotation, shoulder external rotation, shoulder abduction, shoulder adduction, shoulder protraction, and shoulder retraction.

Shoulder Abduction/Adduction: The patient stretches their arm, out straight in front of their body and elevated to shoulder level; They are asked to move their arm sideways away from their body (abduction), back to the original position, and then across, towards the opposite side of their body (adduction).

Shoulder Protraction/Retraction: The patient extends their arm, out straight in front of their body and elevated to shoulder level; They are asked to move their shoulder forward, away from the body (protraction), and then backward, toward the body (retraction).

11.2.2 Real World Challenges

In order to implement a system of this magnitude, that sends sensitive information between a hospital network and an external site, many hurdles needed to be overcome. A few of the major challenges that were encountered, during our implementation, are listed below:

Capture Area: In laboratory conditions, the number of cameras used to capture the person, the position of the cameras, and the size of the area, can all be controlled; this is not possible when dealing with real world situations, like setting up a capture area inside a hospital. The capture area in the hospital was about 2 by $3m$ in size, with uneven lighting. The uneven lighting had to be corrected, by manually calibrating the virtual lighting in the rendered scene. The Kinect V2 sensors that we used, required at least a meter of separation in

order to capture the upper body of the person completely. For this reason, we were restricted to using only one camera on the side. This restricted the overall 3D reconstruction to only one side of the person, instead of a 360° view.

Network Performance: A 3DTI system requires a large amount of data to be transmitted over the network. Each camera, even after compression, generates $200KB$ of data per frame; transmitting this data at $30fps$, would require around $48Mbps$ of bandwidth per camera. The total available bandwidth at the hospital was about $80Mbps$, so only two cameras were used to capture the person at each of the sites, reducing the frame rate to $25fps$.

Protocol Restrictions: Traditionally high frequency haptic data is transmitted over UDP. Since hospital networks do not allow communication over UDP, we had to use TCP instead. Further restrictions made the usage of RTP/RTSP to communicate the 3D data untenable. After going through the firewall restrictions, we determined that the best option was to use SSH tunneling, to communicate all of the data from/to the hospital, to the external site. This heavily restricted our ability to optimize the communication at the network, or packet level.

Security: Since sensitive data was being transmitted over the internet, the data needed to be secure. The use of SSH tunneling ensured the data's security.

Haptic Device: The size of the room, and the use of patients who need rehabilitation, necessitated a smaller device with lower force feedback, for better user comfort and safety.

Display Device: Since the system was used by doctors, and patients who needed rehabilitation, the use of VR displays, such as CAVE, HMD, etc. was avoided; Instead, a $60in$ 3D TV was used.

11.2.3 Motion Mapping

The motions used by the doctor to diagnose the patient, need to be transformed from a physical touch oriented setup, to a touch setup using haptic devices. For this, both the doctor

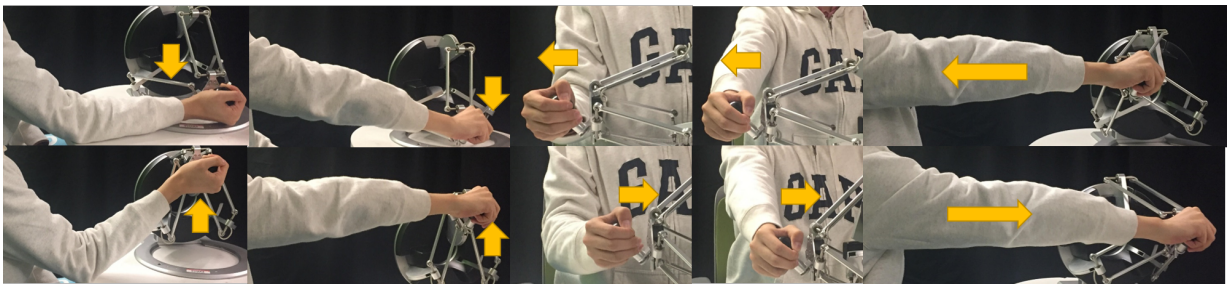

Figure 11.5. Arm motions adapted to the haptic device for diagnosis, shown from top to bottom, left to right: elbow flexion, elbow extension, arm elevation, arm depression, shoulder internal rotation, shoulder external rotation, shoulder abduction, shoulder adduction, shoulder protraction, and shoulder retraction.

and patient were seated in a chair, beside a table containing the haptic device. While seated, each user aligned their shoulder with the handle of the haptic device (neutral position). The motions listed in Section 11.2 were modified, as shown in Figure 11.5, to be used with the haptic device as follows:

Elbow Flexion/Extension: The patient places their elbow on the table and holds the haptic handle; they are asked to bend their arm, pulling the handle toward their shoulder (flexion), and then extend their arm, moving the handle back down toward the table (extension).

Arm Elevation/Depression: Patient stretches their arm out with their elbow extended, straight in front of their body and elevates it to shoulder level while holding the haptic handle; keeping their arm straight, they are asked to move the haptic handle up (elevation), and then down (depression).

Shoulder Internal/External Rotation: While holding their elbow beside their abdomen, the patient bends their arm at the elbow to a 90° position; while holding the haptic handle in their hand, they are asked to move the handle inward (internal rotation) toward their body, and then outward (external rotation) away from their body.

Shoulder Abduction/Adduction: The patient stretches their arm out with their elbow extended, straight in front of their body and elevates it to approximately shoulder

level while holding the haptic handle; keeping their arm straight, the patient is asked to move the handle sideways away from their body (abduction), back to the original position, and then across, toward the opposite side of their body (adduction).

Shoulder Protraction/Retraction: The patient extends their arm out with their elbow extended, straight in front of their body, and elevates it to shoulder level while holding the haptic handle; keeping their arm straight, the patient is asked to move the handle forward (protraction) away from the body, and then backward (retraction) toward the body.

11.2.4 Implementation

The RPDS system was implemented over two geographically distributed sites, as shown in Figure 11.6. Both sites are setup with each user seated at a table with the haptic device, looking at a TV; The doctor is at one site, and the patient is at the other, as shown in Figure 11.1 Each site had a rendering machine Intel i7 5600k @ 4.0 GHz, 32GB DDR4 RAM, NVIDIA GTX 970, connected to a Force Dimension Omega 3 haptic device. The rendering machine renders the scene in 3D, on a $60in$ Samsung 3D TV. Each site also contains a side camera machine Intel Xeon W3530 @ 2.8 GHz, 14GB DDR3 RAM, NVIDIA Quadro 4000. Both of the machines are also connected to MS Kinect V2 RGB-D cameras, to capture the scene. All of the machines on each site are connected by gigabit Ethernet. A machine Intel Xeon W3530 @ 2.8 GHz, 6GB DDR3 RAM, and running CentOS is open to the internet, and acts as the gateway between the two sites.

The 3DTI software is developed in C++ using CUDA 8, BOOST, CGAL, Eigen, GLEW, GLFW, GLM, OpenCV, OpenGL, Speex, TBB, Zlib, and a host of device specific libraries for MS Windows. The 3DTI framework integrates with the Unity3D gaming engine, to provide real-time detailed interactive scenes. Network Time Protocol (NTP) is used to synchronize the clocks of all of the machines, to be millisecond precise.

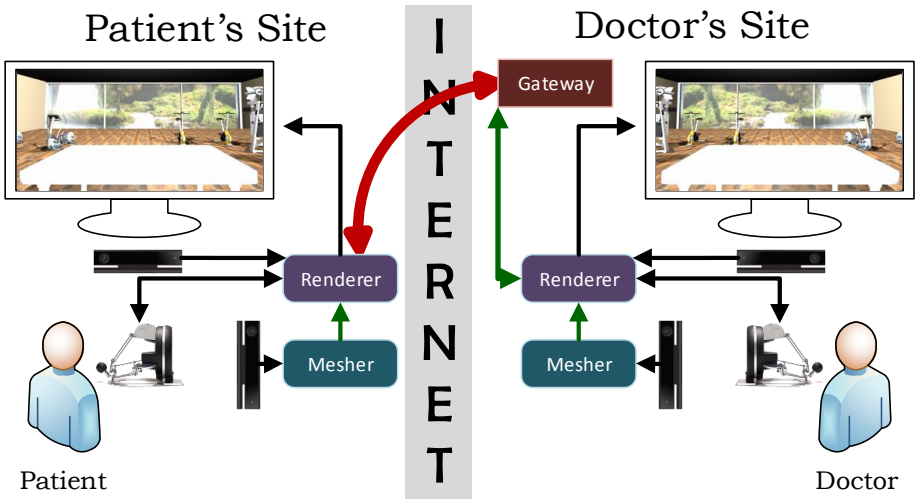

Figure 11.6. The two site setup of RPDS, with the doctor at one site, and the patient at another.

For fast transmission over the network, the mesh information was tightly packed to smaller 16bit floats, and the texture was encoded as a high quality JPEG image. To overcome the issues associated with the network restrictions on the hospital side, a secure gateway machine running CentOS 7 was setup at the university side, for all of the inter site communication. Putty was used on windows to setup the tunnel, with the university gateway machine running SSH daemon based on OpenSSH, this allowed the application to surpass all of the network restrictions at the hospital. The packet compression at the application level, using Zlib, was disabled and the SSH v2 compression for large payloads, using Zlib, was used to compress the packets that were transmitted over the hospital and university network. A separate tunnel was used for large mesh data streams, for optimal performance and flow control. Smaller high frequency haptic data packets were transmitted over an uncompressed tunnel, for faster performance. This multiple tunnel based approach was much faster than using a single compressed tunnel, for all of the transmissions.

11.3 Experiments

The RPDS was developed using an agile approach, allowing for quick feedback from the doctors. After the initial setup, and validation of the system by the doctor, the system was stress tested and benchmarked, to ensure the safety of the users at all times. Latency, motion, and force bounds were established for the safe operation of the haptic device, with communication over the internet. A preliminary trial, using 5 healthy volunteers as patients, was performed to evaluate the real world usage of the system. These sessions were used by both the doctor and system developers to improve the process, modify the motions, update the system, survey questions, etc. After incorporating all of the identified changes, patients with shoulder and elbow issues were recruited to use the system. A total of 15 patients were diagnosed by a doctor in-person, and then another doctor diagnosed the patient using RPDS.

11.3.1 System Performance

The system performance was monitored closely throughout all of the user trials. The round trip ping, between the hospital machine and the university gateway, was around $5ms$. All of the machines at each site are connected to each other using gigabit Ethernet, with about a millisecond of round trip delay. The distributed visual latency approach, described in Chapter 8, cannot be used while using the system, so all of the latency measurements that were carried out during the user trials, are the implicit latencies of the system. For the entire scene, the visual latency is, on average, about $83ms$ more than the measured implicit latency. Both the TV display, and the Kinect V2 sensors, seem to provide stable performance, leading to very little variation. So an approximation of the visual latency, that is experienced by the users, can be calculated by adding $83ms$ to the implicit latency of the system.

The size of the mesh data varies significantly, depending on the contents of the scene, that is being reconstructed to create a mesh. Due to the small room size, the cameras are

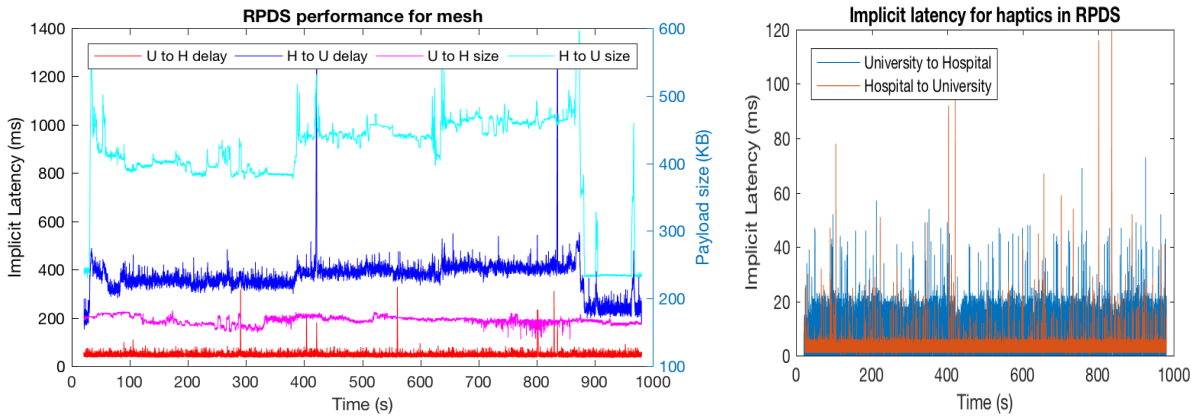

Figure 11.7. The system performance for the mesh (left) and the haptic (right) of the RPDS during a patient diagnosis session between the hospital and university.

placed very close to user in the hospital. While the closeness of the camera provides a better quality capture of the user, it leads to a highly detailed mesh, and results in an increase in the volume of the data to be transmitted every frame. The larger size of the data results in an increased implicit latency, as seen in Figure 11.7.

However, upon further examination, it was found that for the same data, the compression/encryption took about 5 times longer on Putty in windows, than it took on the OpenSSH in CentOS. Distributing the responsibility of transmitting over SSH, to a dedicated Unix machine, can vastly improve the system throughput.

The average haptic data transmission delay for the system was only $3.7ms$, with some large variations at times. A typical diagnosis session using RPDS, takes about 15 minutes; during this span, as seen in Figure 11.7, the implicit latency for the haptic processing remains stable, with a few significant jittery stages. These disruptions lead to delays of over $120ms$ for a single haptic data packet, and a quick recovery by the system, as seen in Figure 11.7. The haptic data aggregation scheme works effectively, even while trasmitting over an SSH tunnel, and eliminates the delay propagation, leading to a quick recovery, even in the cases of large $100 + ms$ jitters.

During all of the user sessions, the haptic transparency performed as expected. The success of the bidirectional haptic transparency is largely due to the relatively low transmission latency between the two sites. To truly test the capabilities of the position based transparency approach, large fixed delays were introduced during transmission, to simulate network conditions. The transparency approach delivers stable forces, for delays as high as $125ms$, when the haptic devices are moved in a similar direction. When the haptic devices move in opposing directions, the forces are no longer smooth, after a delay of $40ms$ is introduced. In situations when nobody is holding onto the haptic device, the force estimation is unstable, even at a latency of $20ms$.

11.3.2 Preliminary Study

An initial study, consisting of 5 healthy volunteers recruited from within the hospital, was used to determine the shortcomings of the system. The doctor showed the volunteers the motions to be performed, and made them repeat them. While interacting with the volunteers, the doctor refined the motions used, the order of the motions, and the necessary questions that were going to be asked to the patients. The users were asked about the appearance, comfort level of using the haptic device, the accurate functioning of the system, the potential of the system, the overall experience, etc. The results of the study are shown in Figure 11.8. Even though the volunteers and doctors were happy with the overall system, they felt that the rendering quality needed to be improved.

For the initial trials, only a single front facing camera was used at each site, to reconstruct the user. While the healthy patients felt that they could see the doctor, and follow the actions effectively, the doctor felt that they were not able to see the patient's arm motions, primarily due to the arm being occluded by the haptic device. For this reason, two cameras were used for the future trials, with one front facing camera, and the other positioned to capture the arm from the side. All of the users preferred seeing both themselves and the doctor, rendered

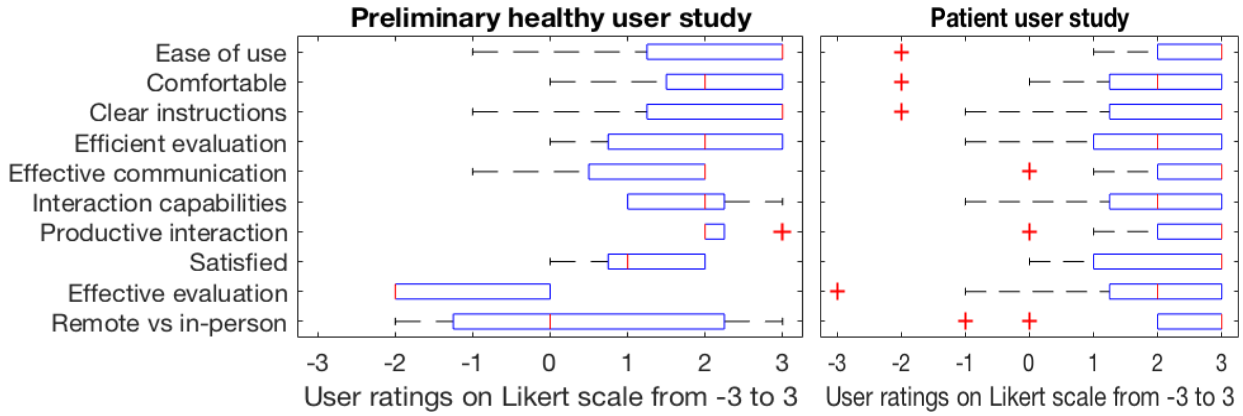

Figure 11.8. User study results for the preliminary study with healthy users (left), and 15 patients with shoulder discomfort (right).

together in the virtual world. The virtual camera was positioned at a 45° angle, to allow both the doctor and the patient to have an unobstructed view of each other, as shown in Figure 11.1. The button on the haptic device was configured to show a close-up view of the patient's hand, as shown in Figure 11.9, giving the doctor access whenever necessary.

The amount of force applied by the doctor and the volunteers on the haptic device, often resulted in the physical displacement of the entire haptic device. To ensure the accurate reconstruction of forces, and the sliding of the haptic device under heavy force conditions, the haptic devices, on both sides, were clamped onto the table, eliminating the movement of the entire haptic device.

All of the users felt that the system was not able to evaluate them effectively, primarily due to the limited amount of force that can be applied by the haptic device, and the motion of the entire haptic device, in situations with large forces. Overall, the user trials served their purpose of providing valuable feedback about the system. The improvements made, based on the feedback, definitely had a positive impact on the future users of the system.

Figure 11.9. The close-up view of the patient, as rendered on the doctor's side, and captured during an RPDS session.

11.3.3 Patient Trials

Patient trials were carried out to verify the real world potential of the RPDS. A total of 15 patients, with shoulder ailments, volunteered to be part of the study. Special care was taken to ensure that a good mix of patients, with varying causes and at different stages of treatment, took part in the study. The patients had suffered from different conditions like: muscular skeleton elbow or shoulder, were in rehabilitation from post shoulder surgery, had a stroke or other neurological condition, had shoulder weakness or shoulder spasticity that was hampering the use of the shoulder.

To compare the usefulness of the RPDS system compared to the in-person diagnosis, a multiple diagnosis approach was adopted. Two doctors were asked to diagnose the range of motion, and maximum isometric strength of the patient, using the 10 exercises described in Section 11.2.1. The first doctor diagnosed the patient in-person, and soon after that, the patient was diagnosed by the other doctor remotely, using the RPDS. The patient described

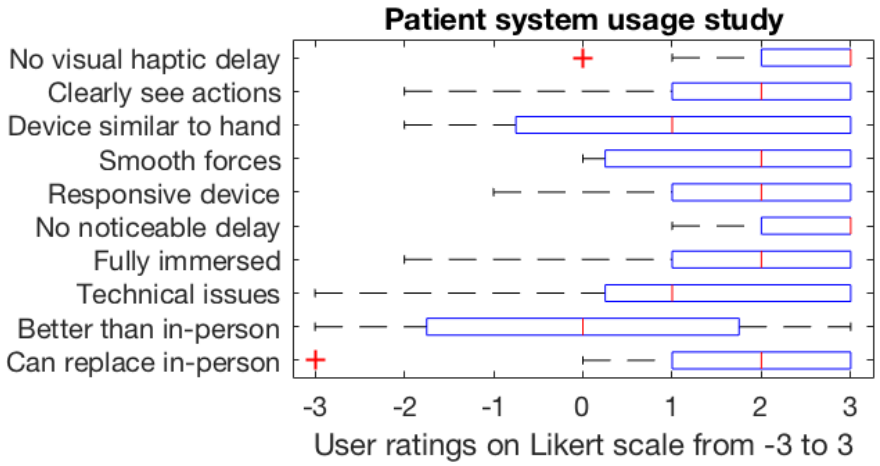

Figure 11.10. User study results for the system specific questions, answered by the 15 patients.

their conditions to both of the doctors, before the start of the session. Each doctor diagnosed the patient based on the 10 shoulder exercises. Both of them noted their diagnosis for the patient, for the range of motion and maximum isometric strength approaches. After finishing the RPDS session, the patient was asked to fill out the detailed questionnaire, shown in Appendix B.

The user experience study results are shown in Figure 11.8. Almost all of the patients reported having a good experience, while being remotely diagnosed by the doctor. Most of the patients felt that the RPDS based diagnosis was as good as the in-person evaluation. The patients showed a higher level of satisfaction with the experience when compared to the preliminary study by the healthy volunteers, indicating that the improvements made to the system after the first study, have improved the entire RPDS experience.

The patients overwhelmingly were happy with the performance of the RPDS, as shown in Figure 11.10. While many felt that the feedback from the haptic device was similar to a person;s hand, some felt that the handle of the haptic device was not very convenient to use. Almost all of the patients felt that the system had the potential to replace the in-person diagnosis, a majority of them felt that it is not presently better than the in-person diagnosis.

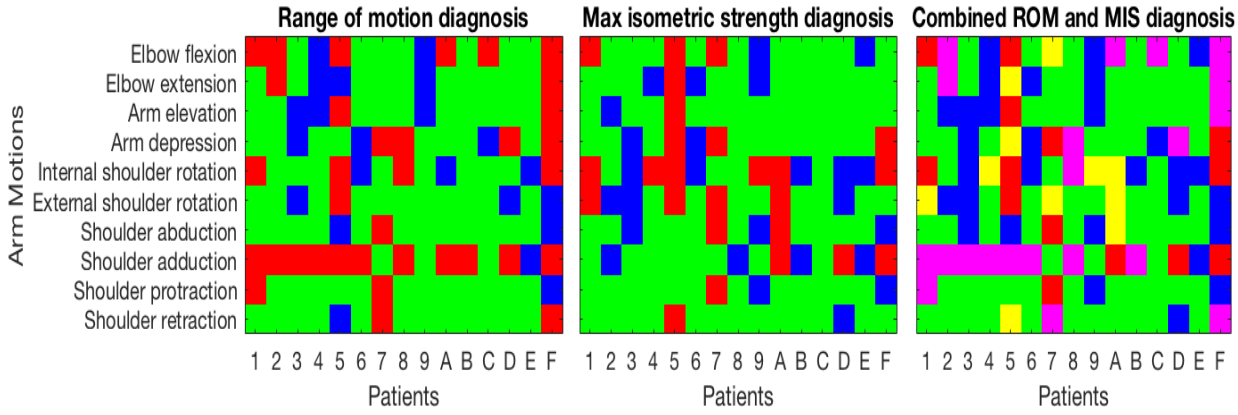

Figure 11.11. The correlation between the in-person and RPDS diagnosis, of the patients in the study, from left to right: range of motion (ROM) and max isometric strength (MIS), green for match, red no match, and blue for match without considering pain; combined ROM and MIS (green means MIS and ROM match, red means none match, blue all match without pain, magenta ROM only match, yellow MIS only match.

There was a high degree of concordance between the in-person and RPDS diagnosis of the patients, by two different doctors, as shown in Figure 11.11. According to the doctors, it is possible that even when diagnosing the same patient in-person, there may not even be a 90% match in diagnosis between the two doctors. So they felt extremely satisfied with the overall consensus, that can be seen in the results.

The range of motion for shoulder adduction needed the patient to move their hand towards their body, and most of the patients could only move their hand a small distance; this caused the doctor to misdiagnose the patient consistently. The doctor felt that a larger size haptic device, with a wider range of motion, was necessary to accurately diagnose shoulder adduction issues of a patient. The doctor was able to accurately diagnose shoulder adduction, for almost all of the patients, using the max isometric strength diagnosis. Overall, between the two methods, the doctor was able to identify all but 7 conditions ($< 5\%$ of all scenarios) accurately, when not considering the pain.

Three patients had difficulty using the system. Along with the shoulder, two of the patients (7 and A) also had issues with their hands, and had difficulty gripping onto the

haptic device, which caused their hands to slip off of the device, resulting in an incorrect diagnosis. The Omega3 haptic device has a smooth ball handle; a different kind of handle, or something to fasten their hand to the handle, would have enabled a better grip and therefore, a better evaluation. The co-operation, between the doctor and the patient, is vital for a proper diagnosis. Patient F did not follow the doctor's instructions, and gripped the haptic handle incorrectly, with the palm of their hand facing upward. Both the in-person, and the RPDS, doctor were not satisfied with their diagnosis of patient F.

11.4 Discussion

The doctors and the patients provided a lot of meaningful feedback while using the system. Some of their interesting observations and insights are as follows:

Haptic Device: Due to the small size of the haptic device, the range of motion is very restricted. A larger device, with a greater range of motion in each direction, would be more useful. The maximum force is equivalent to about $5lbs$, which is easily overcome, even for injured patients. A higher maximum force, of $50lbs$ or more, would be better. The knob handle of the haptic is too smooth, and very hard to hold onto, especially for those patients, who have problems with their hands and/or wrists. Having various types of handles for the haptic, or some kind of gripping/binding to temporarily attach the hand to the haptic handle, would expand its uses, and its ability to help even more people.

More Cameras: The use of only two cameras is not sufficient to generate a complete 360° view of the person, so there are plenty of artifacts, making it difficult to evaluate. Visual noise and artifacts caused by the occlusion of the person, due to the haptic device, make it difficult to accurately visualize what the patient is doing. The inability to move the virtual camera during the session, makes the 3D capture and rendering useless; either a controller based camera motion, or fixed camera positions, is needed for the different actions.

Network Restrictions: The limited bandwidth between the two sites causes the data to be displayed with almost a second of delay, while the haptic data is available within $20ms$. The haptic device becomes unstable, if the delay in haptic data, during a session, is more than 25ms; the current solution is to turn off the forces completely, when any instability is noticed. But a better solution, that ensures accuracy, but not by reducing the force, needs to be developed.

CHAPTER 12

OTHER APPLICATIONS

The framework allows for the creation of i3DTI applications, without the need to write code. The distributed architecture of the framework makes it possible for developers/artists, who are only capable of using a gaming engine, to develop applications without any prior knowledge of the i3DTI framework. Prior to the creation of this framework, there was only a single effort to make any kind of a highly interactive experience for users in a 3D tele-immersion setup. The "I am a Jedi" game (Wu et al., 2010) allowed users to use a real world light saber to interact with each other in a 3DTI setup. The game did not have any scoring, nor any other virtual objects for the players to use. The interaction was largely restricted to one player swinging the light saber at the other player. The framework allows for natural full body interactions between the user and the virtual world. To recreate a game like "I am a Jedi" (Wu et al., 2010), the real world light saber can now be replaced with a much more realistic virtual world light saber.

The ease of creating such applications by using the framework, has allowed many students to create highly interactive games for project work. Using the framework, other applications were created to help solve real world problems, like physical rehabilitation, remote education, etc. Some of these applications (Raghuraman et al., 2012; Venkatraman et al., 2013; Ramalingam, 2016; Desai et al., 2016) are described in the following sections.

12.1 Tennis

A game was created to allow geographically distributed users to play tennis with each other, using the early version of the framework. The two player game had each player positioned opposite each other on a tennis court. Each player was positioned on an independent site, away from the purview of the other player. Each of the sites used a projector to display

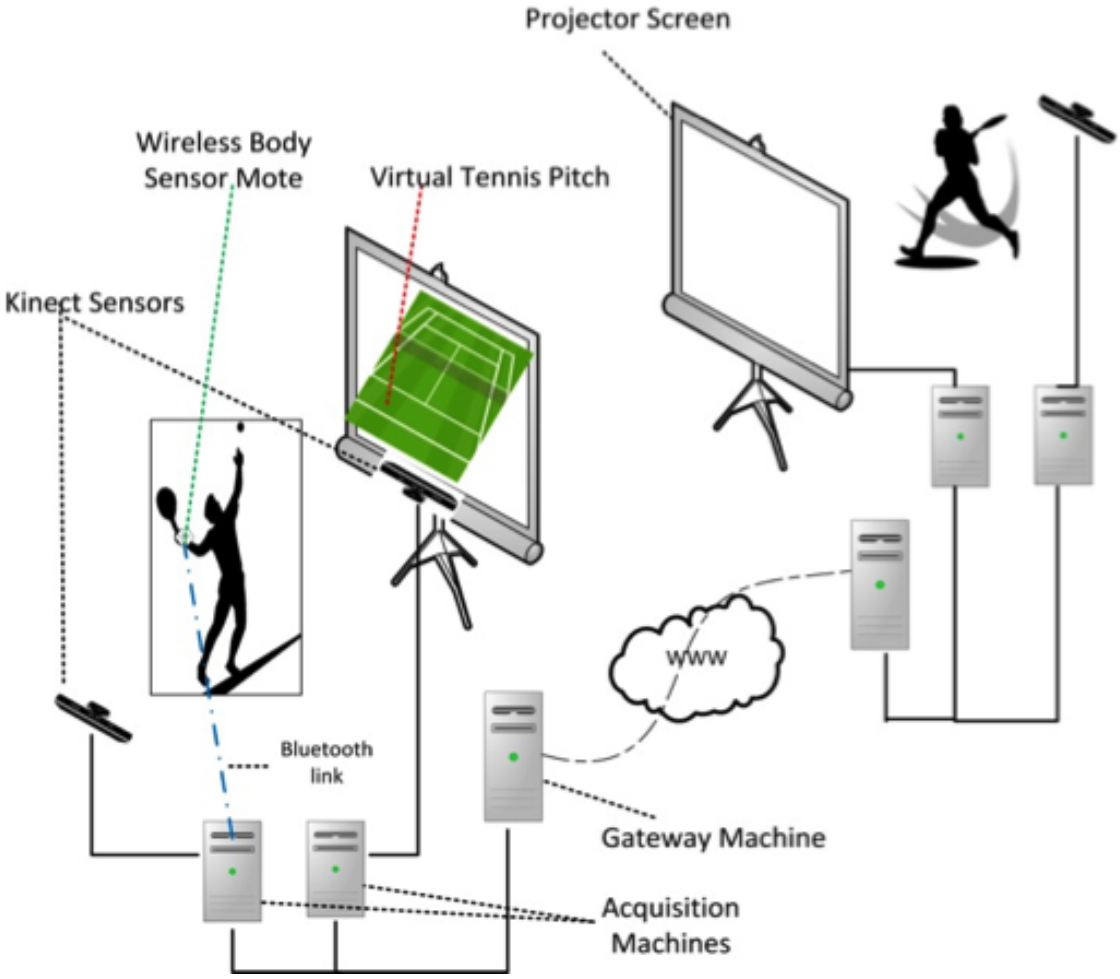

Figure 12.1. An overview of the tennis game setup, using two sites with a player captured using two Kinect V1 cameras on each site, a BSN sensor on each player's hand, and a projector displaying the game.

the player inside the game. The players' motions were scaled up to allow easier navigation throughout their side of the tennis court. Two Kinect V1 cameras were used to capture the player from both the front and the back, as shown in Figure 12.1.

To accurately track the actions of the player, a Body Sensor Network (BSN) sensor capable of 6-axis Inertial Measurement Unit (IMU) was placed on the hand of the player. The BSN sensor tracked the orientation and the accelerations associated with the motions of the hand. A tennis racket was positioned at the player's hand, using the skeleton. The

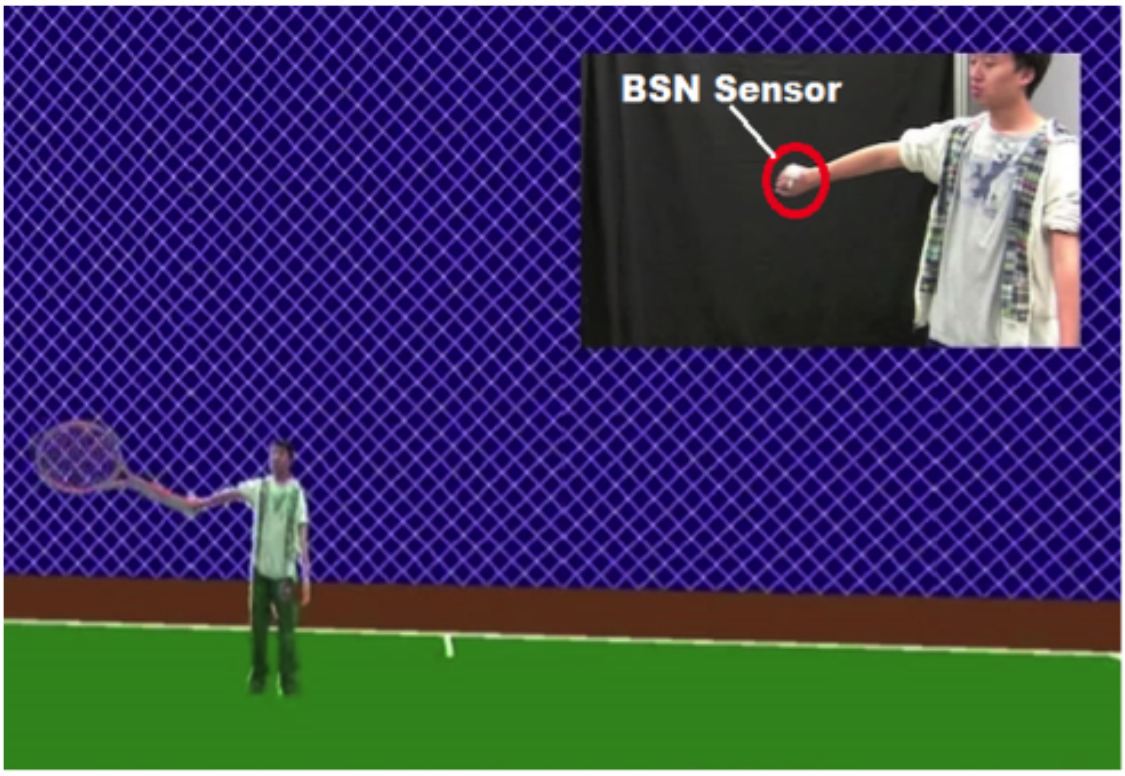

Figure 12.2. The BSN sensor on the player's hand, and the corresponding orientation of the tennis racket in the game.

orientation of the racket was determined by the BSN sensor orientation, as shown in Figure 12.2. The combination of the skeletal hand joint position and the BSN orientation, allowed the player to move the tennis racket and rotate it. Despite the use of the sensor, the gameplay was not very realistic compared to the real world, primarily because of the lack of a real racket.

The tennis racket, the net, and the tennis court, were encapsulated by physics colliders. A ball, with a spherical collider, was introduced for the gameplay. Realistic physics was simulated for: the collision of the ball with the tennis racket, the ball bouncing on the court, and the ball colliding with the net; this allowed the users the ability to serve and return the ball fairly easily.

Figure 12.3. The default third person view of a player playing tennis, showing both the players rendered as a point cloud.

A third person view was used, at both of the sites, to show the user in the virtual world; this allowed the user to see themselves from behind, as shown in Figure 12.3, while also seeing the other player across the net. Open GL rendering was used to render the scene, and the participants. A point cloud was used to show the users in the scene. There was no lighting or shadow effects present in the game, causing user frustration due to their inability to track the ball in the 3D virtual world. All future applications used ample lighting and shadows to allow the users a more realistic rendering in 3D. The shadows give users clearer depth perception, which enables them to track the virtual objects much more accurately. Despite the poor gameplay, that resulted from lower quality rendering, unrealistic motion, and the inability to track the ball, the users were still satisfied with seeing themselves in 3D,

Figure 12.4. The pitcher finishing his pitch, with his arm in front of his leg, and the ball in the air.

and with the potential of the game. These results motivated us to improve the technology and make better applications.

12.2 Baseball

A baseball game was created by students, as a two player experience, with one player as the pitcher and the other as the batter. Each of the players had an individual site, with two Kinect V1 cameras capturing them from the front and back, and a projector rendering the virtual world. Similar to the tennis game, a third person view was used to allow the player to see themselves and the other player, as shown in Figure 12.4. The game was rendered

Figure 12.5. The batter, holding the bat, ready to hit the ball.

in OGRE, with people rendered as meshes instead of a point cloud, leading to a full person view, without holes.

A gesture based control system was created for the player to play the game. When the pitcher is ready to pitch, they bring both hands together in order to make the virtual ball appear in their hands. The pitcher has to wind up with their hand going behind their head, and the ball is released when the hand goes in front of the left leg, as shown in Figure 12.4. The trajectory of the ball, corresponds to the trajectory of the hand, estimated using the position behind the head and the point of release, when the hand goes in front of the left leg. The velocity of the ball is determined by the velocity of the hand between the two reference points (behind head to in front of left leg).

The batter was given a virtual bat, whose handle was positioned in the left hand. The right hand had to be positioned next to the left hand, and controlled the orientation of the bat, as shown in Figure 12.5. By moving both of the hands, like in the real world, the batter is able to move the bat and change its orientation, allowing them to hit the ball.

Figure 12.6. The immersive CPR trainer. (a) shows the real-world scene, (b) shows the visual occlusion problem, (c) shows the virtual world scene.

The game used the prediction method described in Chapter 5 allowing $30 fps$ communication between the sites. The complexity of poses used in Baseball broke the skeletal tracking of the Kinect, leading to inaccurate bat motions and wayward ball releases. This hampered the user experience of the game, requiring better skeletal estimation and tracking, which was later performed, as shown in Chapter 4.

12.3 Cardio-Pulmonary Resuscitation

Cardio-Pulmonary Resuscitation (CPR) is a primary emergency procedure, and instructor-led training courses remain popular. Recent research (Yeung et al., 2011) shows that in self-directed CPR training, performance improves if the trainee is allowed to watch their own actions, since visual feedback can show if their motion correctly mimics that of the trainer. An immersive virtual reality trainer was created, that enabled a trainee to perform CPR training in a 3D virtual world. Our system provides visual and haptic feedback, which helps provide a more realistic approach to learning the motions required to perform CPR. Figure 12.6(a) shows a real world training environment, where the trainee faces a Kinect camera,

and compresses an Omega 3 haptic device in real-time, to perform chest compressions. On the screen, the trainee views his actions as seen in the virtual world; he sees his own "live" image performing CPR on a 3D virtual human, in a supine position, as shown in Figure 12.6(c).

12.3.1 Challenges

The major challenges that need to be overcome by the CPR application are the following:

Visual Occlusion The scene captured by the Kinect has lots of equipment surrounding the trainee, creating occlusion. The trainee is always behind the haptic device and the table, which causes problems capturing and rendering the entire body of the trainee directly, as shown in Figure 12.6(b). Since an Omega 3 haptic device is placed in front of the trainee, and the trainee crosses their hands in front of their body, the captured 3D image will have many white holes.

Chest Deformation The second challenge is to visually show the chest compressions of the virtual human, which is directly related to how much the trainee pushes down on the haptic handle. A fast and stable deformable method is needed to simulate the plausible visual compression, while providing a smooth and stable force feedback back to the trainee.

Synchronization The haptic device and the Kinect generate data at different frame rates. Due to its high data rate of 1KHZ, the haptic information is always faster than the Kinect data. A very fast deformation algorithm, that is used by the system, generates the deformation in real-time, resulting in haptic-based deformation that does not match the Kinect based live model prediction.

12.3.2 Our Solution

The system is setup by positioning the Kinect below the display, with the haptic device in front of the display, and in direct view of the camera. The haptic device is positioned at

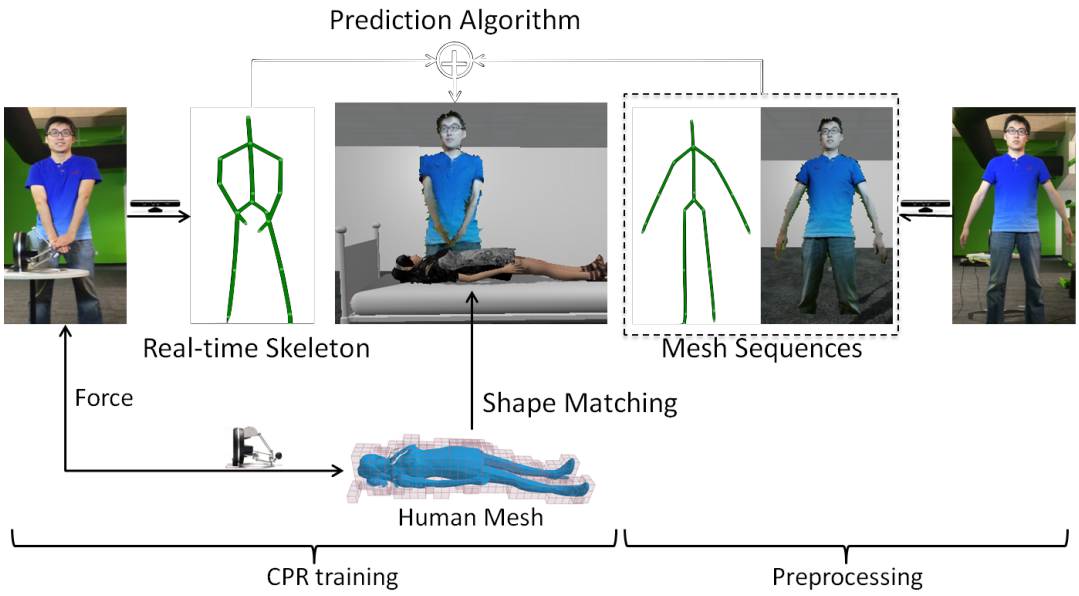

Figure 12.7. The working procedure of the immersive CPR trainer.

both the high and low positions to calibrate the camera, haptic, and virtual space. Once this calibration is done, the system is ready to use. Every time a new trainee steps in, a few frames of the complete person are captured, but typically just one frame is sufficient. The trainee stands behind the haptic device, and performs the CPR compressions on the haptic device, which are rendered by the system on the front display. The system provides both visual and audio clues when an activity is performed incorrectly, such as bending the elbows, or low compressing frequency.

The entire CPR process is based on a prediction algorithm for rendering, and a deformation algorithm for force generation and deforming the virtual model, as shown in Figure 12.7. To lessen the challenge of visual occlusion for rendering, we utilize a prediction based algorithm (Raghuraman et al., 2013); Chapter 5 includes greater detail of this approach. This approach needs a short preprocessing step, where the person is captured. The captured models are then deformed, based on the detected skeleton of the user, in order to predict the user model without the visual occlusion. We begin by capturing a sequence of meshes and skeletons of the trainee, without the haptic device. Once the training procedure begins, the

Kinect's only purpose is to produce the trainee's skeletal joint information. The algorithm regenerates a live mesh model for the trainee using the prediction algorithm.

Chest deformation is solved by using a haptic-enabled deformation model (Tian et al., 2013), which applies a shape matching (Müller, Heidelberger, Teschner, and Gross, Müller et al.) method to simulate the deformation stabily. We start by producing a voxel mesh for the triangle mesh of the human model. Only the chest of the human model will perform simulation. Constraint particle coupling (Tian et al., 2013) can provide smooth force feedback at 1KHz frequency, while the HIP (Haptic Interaction Point) is interacting with the mesh. As shown in Figure 12.7, when the trainee compresses the haptic handle, collision detection is performed between the HIP and the mesh. The force of the collision causes the point of impact to move; this deformation is propagated throughout the model, using shape matching. The force feedback that is provided to the user through the haptic device, is estimated by shape matching while propagating the deformation through the mesh.

To resolve the synchronization issue, we use the calibration that is performed between the haptic space and the Kinect space, to position the trainee in the virtual environment. Given that the trainee is moving the device using both hands, and the position of the HIP is known at every millisecond. we can reposition the hands of the trainee in the virtual space every millisecond. Since the prediction method is based on the position of the joints of the skeleton, the change in the position of the hands results in the corresponding change on the "live" model of the trainee.

Our virtual reality system provides an immersive 3D environment that enables CPR training. With smooth haptic feedback, the trainee feels the force corresponding to each chest compression. To enhance the visual performance, a geometric based method is used to generate the visual change of the human chest under each compression. Through a prediction-based algorithm, we provide a high-fidelity experience where a trainee is placed inside a virtual world, performing CPR on a virtual human.

Figure 12.8. HoloBubble virtual scene from left to right: the design layout, and a virtual rendering that shows the colored bubble generators and mirror setup.

12.4 HoloBubble

The HoloBubble game was created to benchmark the capabilities of the framework. The game uses multiple cameras to capture and render the user in 3D, locally. The delay associated with the capture, processing, rendering, and interaction of the data from each of the cameras, is compounded by the number of cameras used. HoloBubble is a highly interactive game where the user is required to burst bubbles that are being produced by bubble generators. The bubbles are randomly created by the generators at predefined speeds. The bubble generators are arranged in a circle surrounding the user in the center of that circle. Mirrors placed inside the scene allow greater visibility for the players; they are arranged to show the front and sides of the player, as seen in Figure 12.8. Orthogonal reflection probes are used to provide clearer rendering of the player on the mirrors. The game has multiple visual quality and interaction quality modes. The game uses a larger number of cameras for higher visual quality, and a smaller number of cameras for the lowest visual quality. The texture and resolution of the mesh are varied for intermediate visual quality levels. The speed of bubble generation is increased to force faster user response, and slower speeds are used to allow the user to digest the rendered environment. A camera animation was created to revolve around the user, to give them a full 360° view of themselves while playing the game.

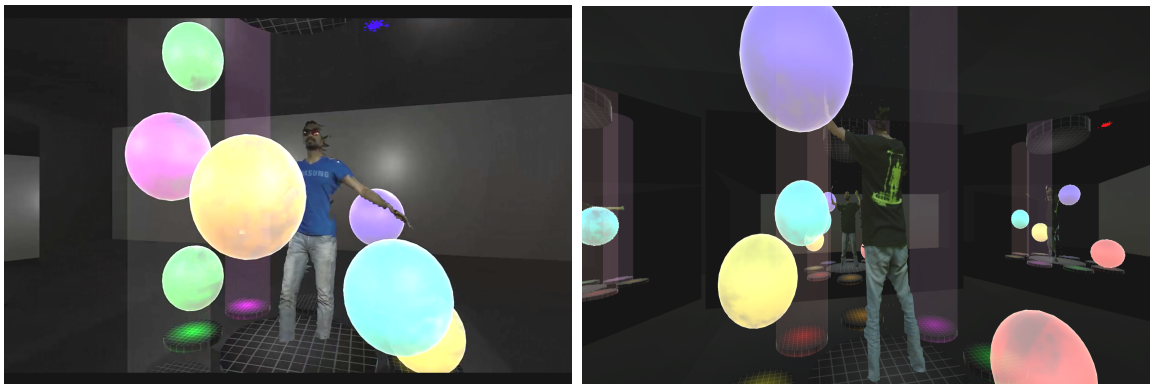

Figure 12.9. From left to right: The front view of the player playing the game, and the rendered view of the player captured from behind, with the mirrors showing the front and side of the player.

Figure 12.8 displays the start screen of the game without the player, the bubble generators are seen on the floor in various colors; each color of the generator indicates the color of bubbles that they produce. The player stands in the center of the bubble generators when they are ready to start the game. The refection of the bubble generators are seen on the mirrors at the front and sides. A long transparent beam of light shows the player where the next bubble will be coming from, helping the player plan their next move. A sample of the HoloBubble game, played by various players, is shown in Figure 12.9. The full body interaction of the game enabled all of the users, irrespective of their gaming experience, to learn and play the game easily. During our evaluation, it was noticed that using reflection probes results in considerably slower rendering on the mirrors, compared to the 3D mesh. The performance of the 3D mesh rendering deteriorated significantly when using more than 6 cameras. Despite this, all of the players were engaged and satisfied by the overall gaming experience of HoloBubble.

12.5 Exergames

Augmented reality based Exergames for rehabilitation are designed and developed using the Unity3D game engine. ShotPut, Bowling, RehabQuiz, and Balance are the 4 Exergames

that were developed, each focusing on a different exercise. A simple strategy was employed: a closed fist was used to grab an object, and an open fist was used to release an object. The open and closed fist detection is performed using the Kinect V2 SDK. Each game has virtual instructions that appear on each start screen. A scoring module calculates the score for each particular game, and provides on-screen updates to motivate the player and make it more challenging. All of the games require the person to be standing or sitting in front of the Kinect, located right under the TV, where they are virtually rendered. The following subsections describe the unique characteristics of each game.

12.5.1 ShotPut

Elbow flexion is an exercise performed to improve the motion of bending and straightening the elbow. The game of *ShotPut* was selected and modified to replicate the elbow flexion exercise. The scene of the game is created to look like a shot put arena, in an open stadium. The game begins with the person standing in the virtual scene just described. The object of this game is to throw the spherical object, the shot, as far away from the player as possible, in the virtual world. By closing their fist, the game recognizes that they are now holding the object (ball or sphere) in their hand, and must stretch that closed hand behind them as far back as possible, over their shoulder; the player then thrusts that hand forward, as quickly and extending it as far as they can, while opening their fist (to indicate that they released the ball). The velocity of the throw is calculated using the change in the position of the closed fist, to the final position of the open fist, with respect to the time it took to go between the two positions. The further the player can extend their arm (both backward and forward), the longer the distance the ball travels from the foul line, and the higher the score the player receives. A virtual camera tracks and follows the ball once it is released. Figure 12.10 shows a person playing the game, from the front; the open hand shows that they released and threw the ball, to where it landed on the ground in front of them. The

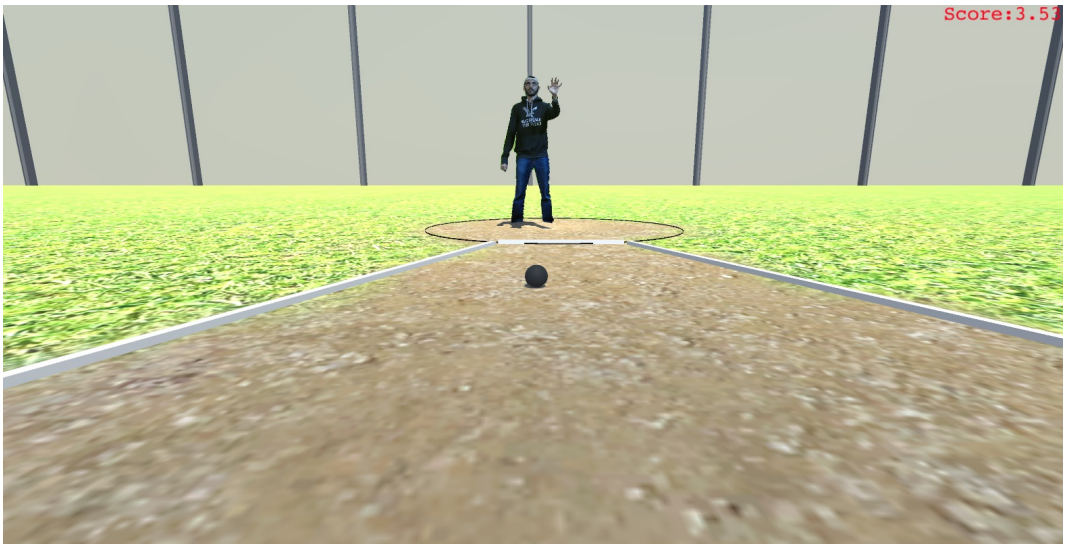

Figure 12.10. The ShotPut game, associated with elbow flexion, shows a person throwing the 'shot' as far as possible, and the distance its thrown from the foul line is how the score is calculated. The score appears in the top right corner.

score in the right hand corner of Figure 12.10 shows the score of the current throw, based on the distance it is from the foul line.

12.5.2 Bowling

Another important exercise that is needed for rehabilitating an elbow is elbow extension. While keeping the elbow straight, the person must stretch their arm out behind them as far as possible, and then bring it to the front. Similar to real world bowling, a virtual 10-pin *Bowling* alley is generated, with the player standing at the end opposite the pins. The player can play this game in a sitting or standing position, and starts the game by closing their fist (holding the ball) with their arm at their side, parallel to their body. The idea of the elbow extension exercise is clearly similar to the motions used in bowling; while keeping the elbow straightened, the person stretches their arm back as far as possible, then swings it in front of them, opening their fist to release the ball. In the real and virtual world, the released ball rolls down the alley and the player's score is calculated by the number of pins knocked

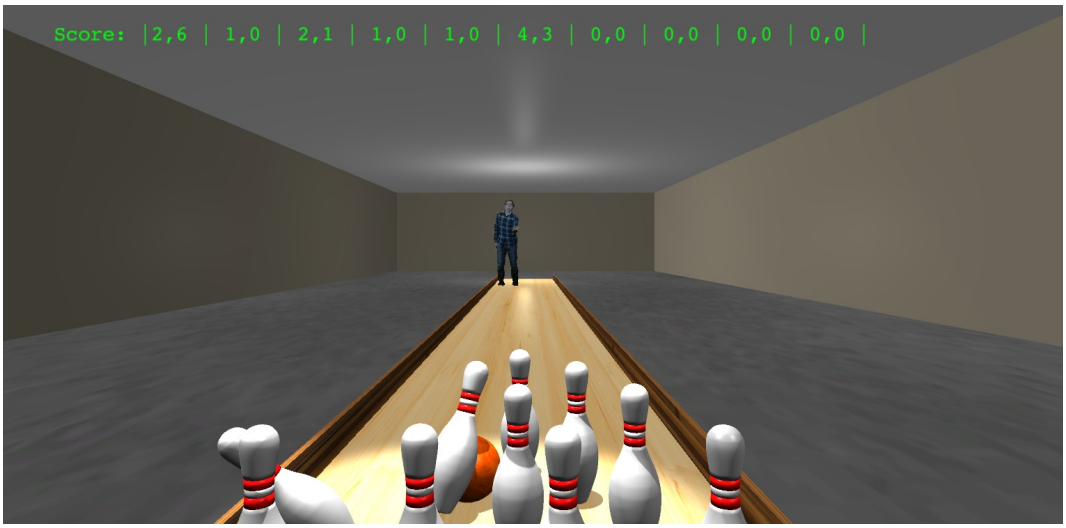

Figure 12.11. The player practices the elbow extension exercise by playing the Bowling game, and earns more points for each pin they knock down.

down by the ball as shown in Figure 12.11. In this virtual Bowling, to keep the players from getting frustrated, the score is calculated only when at least one pin is knocked down. For the ease of use and helpful motivation (needed in rehabilitation scenarios), the bowling alley has the guard rails on the alley to eliminate "gutter" balls, and making it a more fun exercising activity. The force and direction that is applied on the ball is based on the angle at which the person extends the ball backward and then releases it once its swung forward.

12.5.3 RehabQuiz

This Exergame was developed in the hopes of exercising not only the needed physical motion, but adding a mental challenge as well. We created a trivia game, known as *RehabQuiz*, to be played while doing the elbow rotation exercise. This exercise requires the participant to place their elbow and lower arm close to their body (stomach specifically), and then rotating the elbow to move the hand away from the body, while keeping the elbow against the body. Virtually, this game is setup in a dark room, with the player sitting on a chair behind a virtual table; the table contains a deck of trivia cards, along with 3 small bins, sitting to the

Figure 12.12. A person using the elbow rotation exercise while playing the RehabQuiz, a trivia game; this shows how the player picked up a trivia card by closing his hand near the deck of cards, causing a trivia question to appear on the screen.

left of the deck of cards. Each of the 3 bins has an A, B, or C on them, representing the 3 possible answers for each of the trivia cards. After choosing a trivia card, the player must then move their arm to the box with the correct and appropriate answer to the question asked, and then discard that card into that box. This is repeated for each card chosen throughout the game. In playing the game, the person performs an internal rotation of the elbow in order to bring their hand close to the deck of cards to pick a card. The player closes their fist to pick a card from the deck, causing the trivia question from the card to appear on the screen, as shown in Figure 12.12.

Depending on the answer to the trivia question, the player uses an external rotation of their elbow to move their hand to the correct bin (the bin labeled with the letter of the correct answer), where they need to open their fist and drop their card. Depending on their answer, "Wrong" or "Correct" shows up on the table in front of the bins and their score is updated. This game is intended to draw the player's focus to the trivia game, while hopefully taking their mind off of, the possibly painful, but needed exercise they are performing while playing it. This was an attempt to see what kind of activities motivate the participant to

continue doing the exercises they need to do. Finding a way to make stuff we need to do more fun, should always be the goal, especially if it is something that makes us better somehow, like rehabilitation does.

12.5.4 Balance

Lower limb rehabilitation is very important but the majority of the rehabilitation Exergames we created, using virtual reality, focus on upper limbs. We propose *Balance*, a game specifically designed to exercise a player's lower limbs, using the hip abduction exercise. To perform the hip abduction exercise, the person must be standing on only one leg, while stretching the other leg in and out, flexing the hip joint. Since this game can be physically difficult for those with certain lower limb injuries, we decided to design a pleasant and peaceful background for this game. We hoped to set a positive mood with a virtual beach scene, complete with a lighthouse and some trees. The Balance game is started by placing a wooden plank on water, causing it to float and rotate (react) in certain ways, whenever some unstable force is applied to it. The game begins with the player standing on one leg, the left one by default, on the floating wooden plank. The person needs to move, in real life, like they would if they really are floating on that plank.

Since this is a balancing game, the game starts with the player being placed slightly away from the plank's center of gravity, causing it to be unstable; a constant force is being applied to the plank in the direction of the person's center mass, thereby creating an illusion that the person is losing their balance, and will eventually fall into the water. If the angle of the plank comes too close to the water, like greater than 80 or less than 280, the player appears to drown in the virtual world. This is the key motivation to playing this game, to avoid drowning in this virtual world. To avoid drowning, the player must move their right leg away from the body, thereby performing the exercise, hip abduction. As seen in Figure 12.13, by shifting the player's center of mass to the left of their body, and applying the reverse force

Figure 12.13. The Balance game, with the player practicing the hip abduction exercise by standing on one leg, on a plank floating on water.

on the plank, the plank is forced to tilt in the opposite direction. The player's score is based on the duration of time they were able to stay on the plank, without falling into the water. The longer the player stays on the plank, (and alive), the higher their score is for this game. Getting players to do possibly painful or uncomfortable exercises, while engaging their mind in a fun activity may lessen the pain involved in doing the exercise, hopefully just enough to do the exercises necessary for their recovery.

CHAPTER 13

CONCLUSION

Technologies, that make it possible to have real time i3DTI interactions, are very dependent on the hardware used for the capture, processing, and rendering of data. As GPUs continue to double in processing, many of the approaches, that are labeled infeasible for usage with i3DTI systems today, become more likely to provide real-time results. This document provides a comprehensive review and a detailed explanation of how to build an efficient and real-time i3DTI system. Novel approaches for all of the aspects of the i3DTI pipeline are described in the document, starting from calibration.

Portable i3DTI setups are made possible by reducing the time needed to calibrate the cameras in a scene to a couple of minutes, using the ball calibration technique. The segmentation and filtering approaches extract the person and the objects inside the scene, with great precision. The inherent noise, associated with the capture of the scene by the RGB-D camera, results in significant residue, even after filtering the input. The improvements in capture technologies will soon ensure that the input is noise free, like in the case of how color images lead to even better segmentation, and higher quality results. The image based reconstruction approach is able to create a 3D mesh, from even a high resolution depth image, in under a millisecond. The shear speed of the approach, combined with the per camera model of rendering, allows optimization in the processing and transmission that are not possible with a holistic single mesh representation of the user.

In order to maintain coherence between the user and their 3D reconstructed models, in the virtual world while interacting with the system, physical input devices like keyboards and mouse devices are avoided. All of the interaction with the system is controlled using full body natural interactions, while maintaining one to one correspondence between the real and virtual worlds. An efficient low latency multiple Kinect skeleton estimation approach is presented, that can accurately identify the skeleton pose of a person, even if many of the

views are occluded. A visually coherent low fatigue navigation approach is presented, that ensures the user can navigate large virtual spaces from the confines of the small capture area, without any natural motions being seen by the other users of the i3DTI system.

Instead of using compression in transmitting large volumes of data, that are generated every frame by the i3DTI system, innovative approaches that rely on other modalities, like the skeleton information gathered from the depth image, are used to predict the nature of the 3D user mesh data. The skeleton based approaches are shown to require transmission of a few hundred bytes per frame. The prediction approaches process data many times faster than the traditional compression methods, leading to a lower latency rendering of the results. A possible solution to the security challenges introduced by the prediction approaches is provided and a potential area for future research in security of RGB-D feeds is introduced.

Latency estimation approaches used to measure the observed latency at the local site, and between geographically distributed sites, highlight the significant difference between the latency observed by the user of the system, and the latency measured by implicit 3DTI approaches. It is also shown that there is a significant difference in the latency between the reconstruction of an entire scene versus that of an individual user. In highly interactive situations, users do not recognize the visual quality of the models being rendered; since all of their attention is focused on the interaction itself, they notice even small variations in the latency of the system.

We present an immensely flexible i3DTI framework. The ease of development of new i3DTI applications, using the framework, is highlighted by the large number of applications being created by student developers. A real world tele-health application, working over the internet, showcases all of the approaches used by the i3DTI framework. The high degree of correlation between the in-person and remote diagnosis of the patient reaffirms the ability and potential of i3DTI systems, by allowing users to transcend spatio-physical barriers in order to interact with each other in the virtual world, similar to the way they would in the real world.

APPENDIX A

VISUAL QUALITY VS RESPONSIVENESS QUESTIONNAIRE

All the questions were to rate each question as: strongly agree, agree, somewhat agree, neither agree nor disagree, somewhat disagree, disagree strongly, or disagree, except for the additional comments.

- Basic participant profile questions

    1. I am a computer gamer.
    2. I am very experienced with VR technologies.
    3. I have experienced 3D presence before.
    4. I play soccer regularly.

- The following questions were asked for each of the two OVQ and OIQ scenarios.

    Kicking the ball

    1. I was able to find and kick the ball.
    2. There was no noticeable delay while playing.
    3. I was able to kick the ball accurately.

    Kicking the moving ball

    4. I was able to accurately track the position of the ball.
    5. I was able to find the ball throughout the game.
    6. There was no noticeable delay while playing.
    7. I was able to kick the ball accurately.
    8. I was able to kick the moving ball.

9. I was able to move around in the scene.

   Goal Keeping

10. I was able to accurately track the position of the ball.
11. I was able to find the ball throughout the game.
12. There was no noticeable delay while playing.
13. I was able to stop the ball as expected.
14. I was able to move around in the scene.
15. Additional comments

- The overall single player experience questions.

  1. I was completely involved/interested in the game.
  2. I would like to play this game again.
  3. I was comfortable while playing this game.
  4. My avatar was rendered accurately.
  5. I was very interested in the game.
  6. I was satisfied with my overall experience.

- Two player penalty game questions.

  1. Seeing the other player enhances the experience.
  2. There was no noticeable delay in other player's motion
  3. Other player's model was accurate.
  4. Size of the other player was accurate.
  5. I was able to see the other player's expressions.

6. The game was similar to a real world experience.
7. I would like to play the two player game again.
8. The two player game with live avatars is better than a game with virtual models.

- Quality of experience for the entire game single and two player.

1. I was very satisfied with the game.
2. The visual quality of the game was good.
3. The size of the my avatar was accurate.
4. The game was realistic.
5. I was able to learn and play easily.
6. I had to use floor markers to locate virtual objects.
7. The game was similar to real world soccer.
8. I lost sense of space and time.
9. I was completely immersed in the game.

APPENDIX B

REMOTE PHYSICAL DIAGNOSTIC SYSTEM QUESTIONNAIRE

All the questions were to rate each question as: strongly agree, agree, somewhat agree, neither agree nor disagree, somewhat disagree, disagree strongly, or disagree, except for the additional comments.

1. It was easy to use the system.
2. I feel comfortable with the system.
3. The information (verbal instruction) provided with this system is clear.
4. I am able to efficiently complete my evaluation using this system.
5. The system was nice to look at.
6. Overall, I am satisfied with how easy it is to use this system.
7. The device showed the actions of the other person as they were doing it.
8. I could see the other person clearly.
9. I could see the other person's actions clearly.
10. The device resisted and moved like a human hand.
11. The device moved smoothly.
12. The device was very responsive.
13. There was no noticeable delay while interacting with the other person.
14. I could effectively communicate with the other person.

15. I was able to correct any miscommunication while using the system, by effectively interacting with the other person.

16. This system has all the functions and capabilities I expect it to have, to interact with the other person effectively.

17. The features of the system are effective in helping me learn and compete tasks.

18. I believe the virtual interaction using the system was productive.

19. I felt a part of the virtual world shown on TV.

20. There were no technical difficulties or interruptions during the session.

21. Overall, I am satisfied with the use of this system.

22. The system allows me to be physically evaluated as effectively as I would be in a standard in-person evaluation.

23. In comparison with a standard in-person evaluation, it is as easy to interact with, and be examined by, the remote person using this system.

24. Overall, I believe this system is better than a standard in-person evaluation.

25. Overall, I believe this system has potential to replace standard in-person clinical evaluation.

Doctor specific questions

26. It was possible to evaluate the arm strength of the patient using the system.

27. The primary in-person evaluation, and the system based evaluation, provided the same result.

28. I was able to see the limb movement of the patient in order to evaluate them.

REFERENCES

Adams, R. and L. Bischof (1994, June). Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell. 16*(6), 641–647.

Adelstein, B. D., E. R. Johnston, and S. R. Ellis (1996, January). Dynamic response of electromagnetic spatial displacement trackers. *Presence: Teleoper. Virtual Environ. 5*(3), 302–318.

Alexiadis, D., D. Zarpalas, and P. Daras (2013, Feb). Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *Multimedia, IEEE Transactions on 15*(2), 339–358.

Alexiadis, D., D. Zarpalas, and P. Daras (2014, Dec). Fast and smooth 3d reconstruction using multiple rgb-depth sensors. In *Visual Communications and Image Processing Conference, 2014 IEEE*, pp. 173–176.

Arun, K. S., T. S. Huang, and S. D. Blostein (1987, May). Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell. 9*(5), 698–700.

Aspert, N., D. Santa-Cruz, and T. Ebrahimi (2002). Mesh: measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, Volume 1, pp. 705–708.

Aurenhammer, F. (1991, September). Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput. Surv. 23*(3), 345–405.

Auvinet, E., J. Meunier, and F. Multon (2012, July). Multiple depth cameras calibration and body volume reconstruction for gait analysis. In *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, pp. 478–483.

Barbosa, I., M. Cristani, A. Del Bue, L. Bazzani, and V. Murino (2012). Re-identification with rgb-d sensors. In *Computer Vision ECCV 2012. Workshops and Demonstrations*, Volume 7583, pp. 433–442. Springer Berlin Heidelberg.

Beck, S. and B. Froehlich (2015). Volumetric calibration and registration of multiple rgbd-sensors into a joint coordinate system. In *3D User Interfaces (3DUI), 2015 IEEE Symposium on*, pp. 89–96. IEEE.

Beck, S., A. Kunert, A. Kulik, and B. Froehlich (2013, April). Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on 19*(4), 616–625.

Chan, C., C. Yamabayashi, N. Syed, A. Kirkham, and P. G. Camp (2016). Exercise telemonitoring and telerehabilitation compared with traditional cardiac and pulmonary rehabilitation: a systematic review and meta-analysis. *Physiotherapy Canada 68*(3), 242–251.

Chen, K.-T., Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei (2011). Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, New York, NY, USA, pp. 1269–1272. ACM.

Chen, S. and K. Nahrstedt (2013, Dec). Impact of morphing-based frame synthesis on bandwidth optimization for 3dti video. In *Multimedia (ISM), 2013 IEEE International Symposium on*, pp. 211–218.

Chen, X., A. Golovinskiy, and T. Funkhouser (2009, August). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*(3).

Corsini, M., E. Gelasca, T. Ebrahimi, and M. Barni (2007, Feb). Watermarked 3-d mesh quality assessment. *Multimedia, IEEE Transactions on 9*(2), 247–256.

Desai, K., K. Bahirat, S. Raghuraman, and B. Prabhakaran (2015, Dec). Network adaptive textured mesh generation for collaborative 3d tele-immersion. In *2015 IEEE International Symposium on Multimedia (ISM)*, pp. 107–112.

Desai, K., K. Bahirat, S. Ramalingam, B. Prabhakaran, T. Annaswamy, and U. E. Makris (2016). Augmented reality-based exergames for rehabilitation. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, New York, NY, USA, pp. 22:1–22:10. ACM.

Di Luca, M. (2010, December). New method to measure end-to-end delay of virtual reality. *Presence: Teleoper. Virtual Environ. 19*(6), 569–584.

Domiter, V. and B. Zalik (2008, January). Sweep-line algorithm for constrained delaunay triangulation. *Int. J. Geogr. Inf. Sci. 22*(4), 449–462.

Farid, H. (2009, March). Image forgery detection. *Signal Processing Magazine, IEEE 26*(2), 16–25.

Fridrich, A. J., B. D. Soukal, and A. J. Luk (2003). Detection of copy-move forgery in digital images. In *in Proceedings of Digital Forensic Research Workshop*.

Friston, S. and A. Steed (2014, April). Measuring latency in virtual environments. *IEEE Transactions on Visualization and Computer Graphics 20*(4), 616–625.

He, D., D. H. Fuhu, D. Pape, G. Dawe, and D. S (2000). Video-based measurement of system latency. In *International Immersive Projection Technology Workshop*.

Herrera, D., J. Kannala, and J. Heikkilä (2012). Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*(10), 2058–2064.

Hilsmann, A., P. Fechteler, and P. Eisert (2013). Pose space image based rendering. *Computer Graphics Forum 32*(2pt3), 265–274.

Horn, B. K. P., H. Hilden, and S. Negahdaripour (1988). Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA 5*(7), 1127–1135.

Huang, C.-H., E. Boyer, and S. Ilic (2013, June). Robust human body shape and pose tracking. In *3D Vision - 3DV 2013, 2013 International Conference on*, pp. 287–294.

Izadi, S., D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon (2011). 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*, UIST '11, New York, NY, USA, pp. 559–568. ACM.

Keshner, E. A. and R. V. Kenyon (2004). Using immersive technology for postural research and rehabilitation. *Assistive Technology 16*(1), 54–62. PMID: 15357148.

Kowalski, M., J. Naruniec, and M. Daniluk (2015, Oct). Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors. In *3D Vision (3DV), 2015 International Conference on*, pp. 318–325.

Kum, S.-U. and K. Mayer-Patel (2005, May). Real-time multidepth stream compression. *ACM Trans. Multimedia Comput. Commun. Appl. 1*(2), 128–150.

Kurillo, G. and R. Bajcsy (2013). 3d teleimmersion for collaboration and interaction of geographically distributed users. *Virtual Reality 17*(1), 29–43.

Kurillo, G., A. Y. Yang, V. Shia, A. Bair, and R. Bajcsy (2016). *New Emergency Medicine Paradigm via Augmented Telemedicine*, pp. 502–511. Cham: Springer International Publishing.

Ladicky, L., P. H. Torr, and A. Zisserman (2013). Human pose estimation using a joint pixel-wise and part-wise formulation. *2013 IEEE Conference on Computer Vision and Pattern Recognition 0*, 3578–3585.

Lavoué, G. (2011). A multiscale metric for 3d mesh visual quality assessment. *Computer Graphics Forum 30*(5), 1427–1437.

Lewis, J. P., M. Cordner, and N. Fong (2000). Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, New York, NY, USA, pp. 165–172. ACM Press/Addison-Wesley Publishing Co.

Li, B., L. Heng, K. Koser, and M. Pollefeys (2013). A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1301–1307. IEEE.

Lien, J.-M., G. Kurillo, and R. Bajcsy (2007). Skeleton-based data compression for multi-camera tele-immersion system. In *Advances in Visual Computing*, Volume 4841 of *Lecture Notes in Computer Science*, pp. 714–723. Springer Berlin Heidelberg.

Lien, J.-M., G. Kurillo, and R. Bajcsy (2009, November). Multi-camera tele-immersion system with real-time model driven data compression: A new model-based compression method for massive dynamic point data. *Vis. Comput. 26*(1), 3–15.

Maciejasz, P., J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt (2014). A survey on robotic devices for upper limb rehabilitation. *Journal of neuroengineering and rehabilitation 11*(1), 3.

Madgwick, S. O. H., A. J. L. Harrison, and R. Vaidyanathan (2011, June). Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7.

Maimone, A., J. Bidwell, K. Peng, and H. Fuchs (2012). Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Computers & Graphics 36*(7), 791–807.

Mekuria, R., M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar (2013). A 3d tele-immersion system based on live captured mesh geometry. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, New York, NY, USA, pp. 24–35. ACM.

Mekuria, R., M. Sanna, E. Izquierdo, D. Bulterman, and P. Cesar (2014, Nov). Enabling geometry-based 3-d tele-immersion with fast mesh compression and linear rateless coding. *Multimedia, IEEE Transactions on 16*(7), 1809–1820.

Milani, S., M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro (2012, 12). An overview on video forensics. *APSIPA Transactions on Signal and Information Processing 1*.

Miles, H. C., S. R. Pop, S. J. Watt, G. P. Lawrence, and N. W. John (2012). A review of virtual environments for training in ball sports. *Computers & Graphics 36*(6), 714 – 726.

Mine, M. R. (1993). Characterization of end-to-end delays in head-mounted display systems. Technical report, Chapel Hill, NC, USA.

Müller, M., B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. SIGGRAPH '05, pp. 471–478.

of Health, U. D. and H. Services (2015). Summary health statistics: National health interview survey.

Ohl, S., M. Willert, and O. Staadt (2015, Dec). Latency in distributed acquisition and rendering for telepresence systems. *IEEE Transactions on Visualization and Computer Graphics 21*(12), 1442–1448.

Otsu, N. (1979, Jan). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics 9*(1), 62–66.

Pajarola, R., M. Sainz, and Y. Meng (2003). Depth-mesh objects: Fast depth-image meshing and warping. Technical report.

Petit, B., J.-D. Lesage, E. Boyer, J.-S. Franco, and B. Raffin (2009, May). Remote and collaborative 3d interactions. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009*, pp. 1–4.

Popescu, A. C. and H. Farid (2004). Exposing digital forgeries by detecting duplicated image regions. Technical report.

Pulli, K., H. Hoppe, M. Cohen, L. Shapiro, T. Duchamp, and W. Stuetzle (1997). View-based rendering: Visualizing real objects from scanned range and color data. In J. Dorsey and P. Slusallek (Eds.), *Rendering Techniques 97*, Eurographics, pp. 23–34. Springer Vienna.

Raghuraman, S., K. Bahirat, and B. Prabhakaran (2015, June). Evaluating the efficacy of rgb-d cameras for surveillance. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pp. 1–6.

Raghuraman, S. and B. Prabhakaran (2015). Distortion score based pose selection for 3d tele-immersion. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, VRST '15, New York, NY, USA, pp. 227–236. ACM.

Raghuraman, S., K. Venkatraman, Z. Wang, B. Prabhakaran, and X. Guo (2013). A 3d tele-immersion streaming approach using skeleton-based prediction. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, New York, NY, USA, pp. 721–724. ACM.

Raghuraman, S., K. Venkatraman, Z. Wang, J. Wu, J. Clements, R. Lotfian, B. Prabhakaran, X. Guo, R. Jafari, and K. Nahrstedt (2012). Immersive multiplayer tennis with microsoft kinect and body sensor networks. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, New York, NY, USA, pp. 1481–1484. ACM.

Ramalingam, S. (2016). *Importance of interaction in interactive 3D Tele-immersion*. Ph. D. thesis, The University of Texas at Dallas.

Redert, A., M. de Beeck, C. Fehn, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, I. Sexton, and P. Surman (2002). Advanced three-dimensional television system technologies. In *3D Data Processing Visualization and Transmission*, pp. 313 – 319.

Rosten, E. and T. Drummond (2006, May). Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, Volume 1, pp. 430–443.

Salomon, D. (2007). *Data Compression: The Complete Reference*.

Shi, S., W. J. Jeon, K. Nahrstedt, and R. H. Campbell (2009). M-teeve: Real-time 3d video interaction and broadcasting framework for mobile devices. In *ICST International Conference on Immersive Telecommunications (IMMERSCOM)*, Berkeley, CA, USA.

Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake (2011). Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, Washington, DC, USA, pp. 1297–1304. IEEE Computer Society.

Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake (2013). Real-time human pose recognition in parts from single depth images. In R. Cipolla, S. Battiato, and G. M. Farinella (Eds.), *Machine Learning for Computer Vision*, Volume 411 of *Studies in Computational Intelligence*, pp. 119–135. Springer Berlin Heidelberg.

Shuai, L., C. Li, X. Guo, B. Prabhakaran, and J. Chai (2017, Feb). Motion capture with ellipsoidal skeleton using multiple depth cameras. *IEEE Transactions on Visualization and Computer Graphics 23*(2), 1085–1098.

Sielhorst, T., W. Sa, A. Khamene, F. Sauer, and N. Navab (2007). Measurement of absolute latency for video see through augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, Washington, DC, USA, pp. 1–4. IEEE Computer Society.

Staranowicz, A., G. R. Brown, F. Morbidi, and G. L. Mariottini (2013). Easy-to-use and accurate calibration of rgb-d cameras from spheres. In *Pacific-Rim Symposium on Image and Video Technology*, pp. 265–278. Springer.

Steed, A. (2008). A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, New York, NY, USA, pp. 123–129. ACM.

Swindells, C., J. C. Dill, and K. S. Booth (2000). System lag tests for augmented and virtual environments. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, New York, NY, USA, pp. 161–170. ACM.

Tang, Z., G. Rong, X. Guo, and B. Prabhakaran (2010, march). Streaming 3d shape deformations in collaborative virtual environment. In *Virtual Reality Conference (VR), 2010 IEEE*, pp. 183 –186.

Tian, Y., Y. Yang, X. Guo, and B. Prabhakaran (2013, Oct). Haptic-enabled interactive rendering of deformable objects based on shape matching. In *Haptic Audio Visual Environments and Games (HAVE), 2013 IEEE International Symposium on*, pp. 75–80.

Turk, G. and M. Levoy (1994). Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, New York, NY, USA, pp. 311–318. ACM.

Umeyama, S. (1991, April). Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell. 13*(4), 376–380.

Vaillant, R., L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, and M. Paulin (2013, July). Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*(4), 125:1–125:12.

Vasudevan, R., G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Bajcsy, and K. Nahrstedt (2011, June). High-quality visualization for geographically distributed 3-d teleimmersive applications. *Multimedia, IEEE Transactions on 13*(3), 573–584.

Venkatraman, K., S. Raghuraman, and B. Prabhakaran (2013, July). Demo paper: Demonstration of a 3d tele-immersion framework. In *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, pp. 1–2.

Venkatraman, K., S. Raghuraman, Y. Tian, B. Prabhakaran, K. Nahrstedt, and T. Annaswamy (2014, Dec). Quantifying and improving user quality of experience in immersive tele-rehabilitation. In *Multimedia (ISM), 2014 IEEE International Symposium on*, pp. 207–214.

Vezzani, R., D. Baltieri, and R. Cucchiara (2013, December). People reidentification in surveillance and forensics: A survey. *ACM Comput. Surv. 46*(2), 29:1–29:37.

Wang, K., F. Torkhani, and A. Montanvert (2012). A fast roughness-based approach to the assessment of 3d mesh visual quality. *Computers And Graphics 36*(7), 808 – 818. Augmented Reality Computer Graphics in China.

Weber, O., O. Sorkine, Y. Lipman, and C. Gotsman (2007). Context-aware skeletal shape deformation. *Computer Graphics Forum 26*(3), 265–274.

Wu, J., Z. Wang, S. Raghuraman, B. Prabhakaran, and R. Jafari (2014). Demonstration abstract: Upper body motion capture system using inertial sensors. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, Piscataway, NJ, USA, pp. 351–352. IEEE Press.

Wu, W., A. Arefin, Z. Huang, P. Agarwal, S. Shi, R. Rivas, and K. Nahrstedt (2010, Dec). "i'm the jedi!" - a case study of user experience in 3d tele-immersive gaming. In *Multimedia (ISM), 2010 IEEE International Symposium on*, pp. 220–227.

Wu, W., A. Arefin, G. Kurillo, P. Agarwal, K. Nahrstedt, and R. Bajcsy (2011). Color-plus-depth level-of-detail in 3d tele-immersive video: a psychophysical approach. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, New York, NY, USA, pp. 13–22. ACM.

Wu, W., Y. Dong, and A. Hoover (2013, Feb). Measuring digital system latency from sensing to actuation at continuous 1-ms resolution. *Presence 22*(1), 20–35.

Wu, W., Z. Yang, and K. Nahrstedt (2008, June). A study of visual context representation and control for remote sport learning tasks. In J. Luca and E. R. Weippl (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*, Vienna, Austria, pp. 1180–1189. AACE.

Xu, F., Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.-P. Seidel, J. Kautz, and C. Theobalt (2011). Video-based characters: Creating new human performances from a multi-view video database. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, pp. 32:1–32:10. ACM.

Yang, L., L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik (2015, Aug). Evaluating and improving the depth accuracy of kinect for windows v2. *Sensors Journal, IEEE 15*(8), 4275–4285.

Yang, Z., Y. Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. H. Campbell, and W. Yurcik (2006). Real-time 3d video compression for tele-immersive environments. In *Proceedings of Multimedia Computing and Networking 2006*.

Yang, Z., W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy (2010, March). Enabling multiparty 3d tele-immersive environments with viewcast. *ACM Trans. Multimedia Comput. Commun. Appl. 6*(2), 7:1–7:30.

Yang, Z., B. Yu, K. Nahrstedt, and R. Bajscy (2006). A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '06, New York, NY, USA, pp. 14:1–14:6. ACM.

Yang, Z., B. Yu, W. Wu, and K. Nahrstedt (2006). Collaborative dancing in tele-immersive environment. In *in Proc. of ACM Multimedia (MM'06*.

Yeung, J., D. Okamoto, J. Soar, and G. D. Perkins (2011). Aed training and its impact on skill acquisition, retention and performance–a systematic review of alternative training methods. *Resuscitation 82*(6), 657–664.

Yeung, K.-Y., T.-H. Kwok, and C. C. Wang (2013). Improved skeleton tracking by duplex kinects: A practical approach for real-time applications. *Journal of Computing and Information Science in Engineering 13*(4), 041007.

Yoshizawa, S., A. G. Belyaev, and H.-P. Seidel (2003). Free-form skeleton-driven mesh deformations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, New York, NY, USA, pp. 247–253. ACM.

Zhang, Z. (2000, November). A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. 22*(11), 1330–1334.

Zhou, Z., X. Chen, L. Zhang, and X. Chang (2011, March). Internet-wide multi-party tele-immersion framework for remote 3d collaboration. In *VR Innovation (ISVRI), 2011 IEEE International Symposium on*, pp. 183–188.

BIOGRAPHICAL SKETCH

Suraj Raghuraman was born in Madras (Chennai), India. Suraj spent most of his childhood breaking, then fixing, stuff around the house. It seemed only natural for him to continue on in the same way as he grew up; but by switching his attention and focus to fixing computers, he ended up saving his parents a lot of money in the process. Suraj had an adventurous life moving often and going to schools all over India. He attended MVIT, Bangalore, where he earned a Bachelors of Engineering in Information Science, in 2005. In his last years at MVIT, he began working at Subex as a Fraud and Risk Management engineer. Suraj became bored quickly with his job at Subex; so in 2007, he made the decision to leave his country and come to the US to continue his education at UTD. He completed his Master's degree in Computer Science in 2008. It didn't take long for Suraj to realize that working a full-time job for someone else wasn't going to make him happy. In 2009, he decided to strike out on his own and started his own company, Mobiweb Inc., a place where he can strive to make his motto of "Smart and Lazy" a reality.

CURRICULUM VITAE

Suraj Raghuraman

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: suraj@utdallas.edu

Educational History:

B.E., Information Science, Sir M. Visvesvaraya Institute of Technology, India, 2005
M.S., Computer Science, The University of Texas at Dallas, 2008

Interactive 3D Tele-Immersion
Ph.D. Dissertation
Department of Computer Science, The University of Texas at Dallas
Advisor: Dr. Balakrishnan Prabhakaran

Anomaly Detection from Aviation Safety Reports
Masters Thesis
Department of Computer Science, The University of Texas at Dallas
Advisor: Dr. Vincent Ng

Employment History:

Director, Mobiweb inc, January 2009 – present
Software Developer, Subex Limited, January 2005 – July 2007