ON 3D CONTENT MANIPULATION: SIMPLIFICATION,

MODIFICATION AND AUTHENTICATION

by

Kanchan Anil Bahirat

APPROVED BY SUPERVISORY COMMITTEE:

Balakrishnan Prabhakaran, Chair

Ryan McMahan

Alvaro Cardenas

Sriraam Natarajan

Dedicated to my Aai and Baba

ON 3D CONTENT MANIPULATION: SIMPLIFICATION, MODIFICATION AND AUTHENTICATION

by

KANCHAN ANIL BAHIRAT, BE, MTech

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

December 2018

ACKNOWLEDGMENTS

To begin with, I am deeply grateful to my parents Asha and Anil Bahirat. Without their struggle, encouragement and support at various stages of my life, I could not have been where I am today. It is they who saw this dream for me.

I also owe a lot to my advisor, Dr. B. Prabhakaran, for his continuous support and motivation. Especially for believing in me when I was struggling during the initial phase of my PhD. I would like to sincerely thank Dr. McMahan and Dr. Cardenas for the informative discussions and guidance throughout my research, and Dr. Natarajan for serving on my committee. Mainly, various discussions with Dr. McMahan has helped me to learn critical and analytical thinking which is the essence of PhD research.

I would like to thank all those who have contributed to this work in different ways. Mainly,

- Dr. Thiru Annaswamy, thanks for his guidance and advise regarding the phantom limb project.
- Dr. Gargi Raval and Mitchell Knoll for helping us to make the Mr.MAPP project feasible and useful for the patients with phantom pain.
- Suraj, thanks for always being there to guide me whenever I came across a stumbling block in the research path. I will always remember all insightful discussions and definitely the time when he is calm during the last moments of deadlines when I am completely in the panic state.
- Arvind for his motivational talks for research, photography, music and of course food.
- Yuan for his advise on my career paths.
- Rittika, for being my friend, supporter, critique, guide and partner in crazy activities. Thank you for playing so many roles, it has enriched my life at UT Dallas.

- Kevin, thanks for helping me in emergency situations during various demos and motivating me to focus on career building and to enjoy the life outside the lab.

- Shanthi for being a role model balancing both research and family as well as for all the helpful discussions. I will always remember our visit to Taipei for ACM MMSys 2017 conference.

- Barbara for all the pep talks spanning a variety of topics starting from deep learning, TensorFlow, gardening to my next most critical personal adventure.

Finally, I would like to express my sincere gratitude to my husband, Bhushan. Without him, I would not have been able to realize this dream. He has been a constant source of support and motivation throughout the many long years spent working on this dissertation. For all of his efforts including Austin to Dallas drives during severe weather, various attempts of cooking, always being there for me and for many more things, I feel deeply indebted to him.

October 2018

ON 3D CONTENT MANIPULATION: SIMPLIFICATION,

MODIFICATION AND AUTHENTICATION

Kanchan Anil Bahirat, PhD
The University of Texas at Dallas, 2018

Supervising Professor: Balakrishnan Prabhakaran, Chair

With the rapid development in the area of computer graphics and depth sensing technologies, various tools have emerged for 3D content creation such as RGB-D cameras (e.g., Microsoft Kinect, RealSense), LiDAR sensors, CAD tools etc. By employing these tools, creating and refining the 3D data has become viable which in result has enabled numerous mixed-reality applications in the domain of healthcare, virtual training, collaborative visualization, vehicle automation and crime scene reconstruction. Although the 3D data finds application in diverse areas, deploying an immense 3D content across various platforms is still challenging. Besides, due to applicability in sensitive areas, scrutinizing the vulnerability of 3D data has also become crucial. Motivated by these challenges, this dissertation presents a set of novel approaches for 3D content manipulation mainly focusing on making it effective across different platforms and reliable for various sensitive applications.

Specifically, I have considered three research tasks. The first task focuses on *"3D content simplification for multi-platform rendering"* that includes 1) designing a high-fidelity mesh simplification algorithm QEM_{4VR} enabling mobile virtual reality (VR) and 2) a real-time curvature sensitive surface simplification CS^3 using depth images. Among these, QEM_{4VR} applies curvature based boundary preservation for generating a high-fidelity, low-poly version of 3D meshes in an offline manner. While CS^3 provides a real-time, curvature adaptive

surface simplification from depth images for enabling mixed reality applications. We also studied the utilization of objective perceptual quality metrics to evaluate 3D mesh simplification algorithms that motivates us to formulate a Just Noticeable Difference based subjective analysis.

The second task is aimed at *"3D content modification for virtual therapies"* that explores the applicability of 3D data in managing phantom pain as well as for a virtual enhancement providing the positive reinforcement during virtual therapy. Mixed reality-based framework developed for managing phantom pain (Mr.MAPP) creates a realistic and effective illusion of a phantom limb in case of both upper and lower limb amputation. The system is evaluated by subject matter experts (SMEs) and has been enhanced for conducting the patient trial.

The last task is *"3D content authentication for secure usage"* by performing thorough studies on 3D data forensics for both long and short range sensors such as LiDAR and Microsoft Kinect respectively. Keeping the application of 3D data in self-driving cars, we also propose a framework, ALERT (Authentication, Localization, and Estimation of Risks and Threats), as a secure layer in the decision support system in the navigation control of vehicles and robots. Various experimental results demonstrate the effectiveness of the proposed ALERT for ADAS (Advanced Driver Assistance System).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

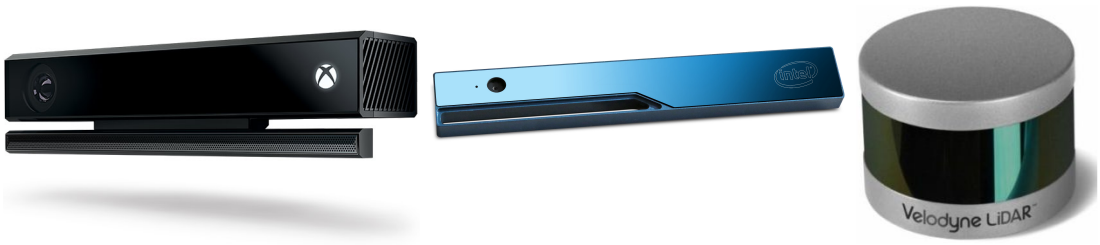
CHAPTER 1

INTRODUCTION

1.1 Motivation

With recent advances in the depth sensing technology, various depth sensors such as LiDAR, RGB-D cameras (see Figure 1.1) are widely available in the market. LiDAR sensors have long-range, millimeter precision and mostly employed in outdoor settings. While RGB-D cameras such as Microsoft Kinect, RealSense are short range, low-cost, available off-the-shelf and usable in indoor environments. Using this wide-range of depth sensors, it has become possible to quickly generate a complete 3D reconstruction of an object or an entire scene. Besides using depth sensors, 3D content can also be generated by software such as Maya, CAD. Equipped with the ability of a rapid 3D content generation, a numerous 3D data in various formats is created, refined and utilized.

Simultaneously, due to the commercial introduction of mobile head-mounted displays (HMDs) such as the Samsung Gear VR, Microsoft Hololens and Oculus Rift, various mixed-reality based applications utilizing 3D data have emerged which find potential utility in the domain of health care, virtual training, collaborative visualization and 3D Tele-immersion (Kreylos, 2005; Arworks, 2015; Microsoft, 2016) (See Figure 1.2). These real-time applications need to balance visual quality, rendering latency, power consumption and battery life to achieve an optimal immersive experience (Microsoft, 2015). However, accomplishing these applications has two prime hurdles: 1] Generally, the 3D content is extensive in size compared to their 2D counterpart. For example, typical RGB-D cameras such as Microsoft (MS) Kinect generates a dense depth data consisting of maximum 300K vertices @30 fps. 2] mobile HMDs are severely constrained by limited graphics processing units (GPUs), which are over an order of magnitude slower than desktop-based GPUs (Boos et al., 2016). Further, even if the input is a compressed depth stream obtained using transmission standards



(a) Microsoft Kinect (b) Intel's RealSense (c) Velodyne LiDAR

Figure 1.1: Short (Microsoft Kinect, RealSense)and long range (LiDARs) 3D sensors.

(a) (b)

Figure 1.2: a) VR training developed by AiSolve (Metry, 2017) and b) Mixed-reality for phantom pain from Multimedia lab of UT Dallas.

such as 3D-HEVC (Tech et al., 2012), the uncompressed stream still gives a dense sampling. Hence, matching the limitation of handheld mobile VR devices and rendering engines while maintaining a realistic, immersive experience is a challenging task. One possible approach to mitigating the GPU limitations of mobile HMDs is to reduce the overall number of polygons that the GPU must render in real-time. Typically, mesh simplification algorithms can be used to rapidly create low-poly versions of pre-existing, high-poly 3D models. However, many existing mesh simplification algorithms are not able to adequately handle manifold meshes with boundaries and non-manifold meshes, which are common attributes of many 3D models.

Although, various mixed and virtual reality applications are developed for virtual therapies, different challenges in deploying these systems for specific tasks such as managing phantom pain, providing positive reinforcement are still not addressed. For example, among a wide spectrum of treatments developed for alleviating phantom limb pain includes virtual reality-based methods. Most of the virtual reality-based methods rely on 3D CAD models of the virtual limb, animating them using the motion data acquired either from patient's existing anatomical limb or myoelectric activity at patient's stump (of the amputated limb). Since motion activity is typically captured using body sensors (Electromyography, EMG, or inertial sensors), these methods are considered as invasive approaches. Further, in the case of virtual reality-based methods, the dependency on the pre-built 3D models degrades the immersive experience due to a mismatch in the skin color, clothes, artificial and rigid look and misalignment of the phantom limb.

Along with health care, education and entertainment domain, the 3D content also find applicability in various sensitive domains such as surveillance, crime scene reconstruction, autonomous vehicle control etc. Many of the vehicle automation and robot navigation applications involve remote guidance - either for safety or for the task performance - of these vehicles and robots. As described in the later section, the ability to manipulate 3D content, although it is beneficial for multi-platform rendering and virtual therapies, it exposes vulnerabilities of using 3D data. Considering the security risks associated with the improper behavior of these applications, it has become crucial to authenticate the 3D data that highly influence the decision making in such applications.

1.2 Questions Addressed

Keeping these challenges in mind, we are mainly addressing the following questions in this dissertation:

1. How to handle limited processing of mobile HMDs?

2. How to design a mesh simplification algorithm that will generate high-fidelity, low poly meshes while preserving boundaries of 3D objects?

3. Are the existing perceptual distance metrics sufficient to capture the 3D mesh quality perceived by humans?

4. How to develop a fast and real-time surface simplification algorithm facilitating the interactive, real-time mixed reality applications?

5. How to create a realistic illusion of a phantom limb by 3D content manipulation for phantom limb pain management?

6. Is 3D data vulnerable to security threats?

7. Is it possible to detect any manipulation/attack on 3D data?

8. Is it possible to create a complete framework that acts as a secure layer for decision support system in the remote navigation of vehicles/robots?

1.3 Dissertation Objectives

Figure 1.3 describes the overall flow of the dissertation. The primary objectives of the dissertation are to study and develop novel methods of the 3D content manipulation such as:

- 3D content simplification for multi-platform rendering
- 3D content modification for virtual therapies
- 3D content authentication for secure usage

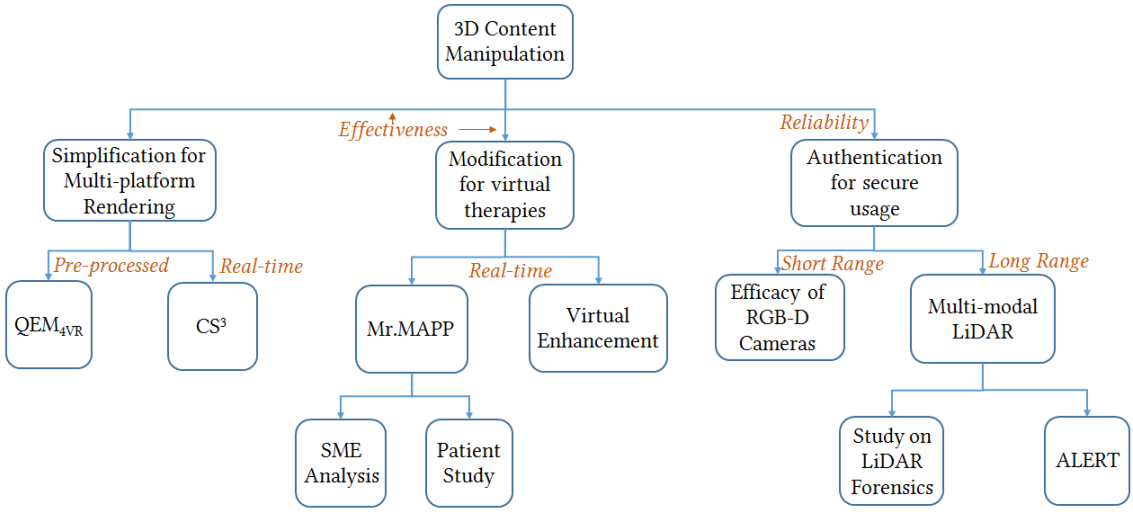


Figure 1.3: The overall flow of the dissertation.

1.3.1 Simplification for multi-platform rendering

To make the 3D content effective across various platforms, we opt for an automated approach of 3D mesh simplification. We present QEM_{4VR}, a high-fidelity mesh simplification algorithm specifically designed for VR. This algorithm addresses the deficiencies of prior quadric error metric (QEM) approaches by leveraging the insight that the most relevant boundary edges lie along curvatures while linear boundary edges can be collapsed. Additionally, our algorithm preserves key surface properties, such as normals, texture coordinates, colors, and materials, as it pre-processes 3D models and generates their low-poly approximations offline.

Although QEM_{4VR} generates high fidelity, low-poly models, similar to traditional mesh and point simplification algorithms, it also needs a high execution time and is inadequate for real-time applications. For examples, in typical real-time applications based on RGB-D cameras, the 3D data consisting of approximately 300K vertices is generated @30 fps. In these applications, all the data must be processed at least @10 fps for unnoticeable delay or lag, i.e., the processing must be done in less than 100 ms (Wu et al., 2011). However, most of the traditional mesh and point cloud simplification algorithms take a couple of seconds to process each frame.

To alleviate this issue, we introduce a depth image-based approach to sparsely sample a surface for real-time mesh generation and visualization. We propose a Curvature Sensitive Surface Simplification - CS^3 operator that assigns an importance measure to each point in the depth image, based on the local curvature. Further, it applies an importance-order based restrictive sampling to generate a sparse representation that retains the overall shape as well as the finer features of the object. We also modify the 2D sweep-line based constrained Delaunay triangulation to generate 3D meshes from the sparse point sampling obtained using CS^3. Additionally, the proposed approach preserves key surface properties, such as texture coordinates and materials.

1.3.2 Modification for virtual therapies

To address the limitation of existing traditional and virtual mirror therapy, we propose a novel Mixed Reality based system for MAnaging Phantom Pain (Mr.MAPP), utilizing off-the-shelf RGB-D cameras such as Microsoft Kinect V2 to capture and generate a live 3D model of the patient in real-time. An illusion of the virtual limb is crafted in real-time by mirroring the patient's symmetric anatomical limb in the captured data with the help of various computer vision and graphics techniques. Along with that, a phantom limb skeleton is also generated in real-time to enable interaction with virtual objects.

1.3.3 Authentication for secure usage

At this end, we perform forensic studies for a 3D data captured using both (i) short-range sensors such as Microsoft Kinect and (ii) long-range sensors such as LiDAR. For short range sensor data, we examined in detail the possibility of creating malicious 3D data stream using simple processing. We also presented a forgery detection method for exposing such malicious data streams. For long-range sensors, (i) we identified three possible approaches for attacks on the LiDAR data that do not need additional commodity hardware, (ii) we also presented

two novel algorithms for detecting such forgeries in LiDAR data in case of different types of attacks.

In the context of self-driving cars/robots and their remote navigation, we propose a framework, ALERT (Authentication, Localization, and Estimation of Risks and Threats), as a secure layer in the decision support system used in the navigation control. ALERT tamper-proofs 3D LiDAR data by employing an innovative mechanism for creating and extracting a dynamic watermark. Next, when tampering is detected (because of the inability to verify the dynamic watermark), ALERT then carries out cross-modal authentication for localizing the tampered region. Finally, ALERT estimates the level of risk and threat based on the temporal and spatial nature of the attacks on LiDAR data. This estimation of risk and threats can then be incorporated into the decision support system used by ADAS (Advanced Driver Assistance System).

1.4 Contributions

The main scientific contributions of my research include:

- A novel mesh simplification algorithm utilizing the QEM infrastructure that uses a curvature-based boundary preservation approach to maintain key boundary edges without sacrificing necessary surface edges. This curvature-based approach avoids creating gaps at boundaries and holes within surfaces. It yields more-accurate, low-poly meshes, along with preserving key surface properties, such as texture, normals and material properties.

- A details study on investigating an applicability of existing perceptual mesh quality measurements for evaluating mesh simplification algorithms designed for non-manifold meshes with boundaries.

- A real-time depth image based sparse sampling method for 3D surfaces, which is sensitive to the underlying surface curvature. It maps the 3D surface simplification problem to the 2D domain by utilizing the depth image. The sampling method and modified triangulation inherently eliminate the sensor noise at object boundaries as a byproduct. The proposed method can be easily extended to obtain distance-dependent simplification of the complete scene. It allows a user-controlled sparseness that can potentially mitigate the limitations of handheld VR devices and rendering engines.

- A novel mixed reality based system for MAnaging Phantom Pain (Mr.MAPP) which creates a phantom limb using augmented virtuality. It provides a cost-effective solution that is simple and easy to use for the relief from phantom limb pain. It is a non-invasive approach that provides a more realistic and natural representation of the phantom limb that matches the person's skin tone and clothing. It facilitates an interaction-enabled, lifelike, and smooth movement of the phantom limb.

- Animation based virtual enhancement is developed as a key feature for immersive exergames (exercise and gaming) that provides a positive reinforcement and motivation to perform tasks in the AR space that are not possible in the real world.

- A 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. A forensic approach for exposing the 3D data stream manipulations is presented that employs a block-based depth noise evaluation approach to detect manipulations in the depth stream.

- A detailed study of possible forgery attacks on LiDAR data is performed that outlines two novel forensic approaches for LiDAR data (As per our knowledge, this is the first attempt to address LiDAR forensics). The proposed forensic algorithms are effective for

specific types of forgeries, a forensic approach handling the wide spectrum of forgeries is necessitated. This work creates awareness about avoiding the blind usage of LiDAR data in critical applications and motivates to explore forensic of LiDAR data as an emerging research area.

- A novel framework, ALERT, that acts as a secure layer in the decision support system used in remote vehicle/robot navigation. It utilizes dynamic watermarking scheme for tamper-proofing 3D LiDAR data. An additional level of scrutiny is provided by cross-modal authentication and risk factor assessment that analyze the 3D LiDAR data thoroughly.

The corresponding set of publications are listed in curriculum vitae.

1.5 Dissertation Outline

This dissertation is logically divided into three parts. Part I includes Chapter 2 and Chapter 3. Chapter 2 describes various challenges in using same 3D content across various platforms and proposed mesh simplification algorithm for creating high-fidelity, low-poly 3D meshes offline. Chapter 3 outlines the requirement of real-time surface simplification and details of the proposed curvature sensitive surface simplification processing depth images in the real-time. Part II includes Chapter 4 describing the proposed Mr.MAPP framework and its extension for Lower limb amputation. It also sketches the patient study details along with the description of the virtual enhancement algorithm. Part III consists of Chapter 5, Chapter 6 and Chapter 7. Chapter 5 evaluates the efficacy of RGB-D cameras, Chapter 6 presents a study on LiDAR forensics. While Chapter 7 describes a complete framework adding a secure layer in the decision-making system for reliable remote vehicle/robot navigation.

PART I

SIMPLIFICATION FOR MULTI-PLATFORM RENDERING

With the ever increasing range of available 3D sensors and the commercial introduction of various virtual reality and augmented reality head mounted displays, diverse set of virtual and mixed reality applications are emerging. However, these applications are severely constrained with the magnitude of 3D content and limited processing power of VR/AR head mounted displays. With this motivation, we suggest adaptively simplifying the 3D content to cater across various platforms.

Chapter 2: Typically, the 3D content used in majority virtual realty application consists of 3D meshes that are designed and built offline. To utilize the same 3D environemantthat is deployed in the desktop environment in head mounted displays, the 3D meshes must be artistically reduced to low-poly versions which can be a tedious task. Hence, we develop an automated, iterative mesh simplification algorithm that performs curvature based boundary preservation to yield high fidelity low-poly meshes.

Chapter 3: Although, the iterative mesh simplification algorithms can be very effective, they are not efficient in reducing the live 3D data captured using depth sensors such as Microsoft Kinect in real-time. To address this issue, a curvature sensitive surface simplification provides an efficient solution that simplifies the live 3D depth data in real-time achieving the desired frame rate of 10 fps for mixed reality applications.

CHAPTER 2

A BOUNDARY AND TEXTURE PRESERVING MESH SIMPLIFICATION ALGORITHM FOR VIRTUAL REALITY[1]

2.1 Introduction

Mobile virtual reality (VR) has garnered a lot of public attention during the past couple years. The main reason for this attention has been the commercial introduction of mobile head-mounted displays (HMDs) that run on smartphones fitted into head-based peripherals, such as the Samsung Gear VR and Google Cardboard. These mobile HMDs have several advantages over traditional desktop-based VR systems. First, because these systems are self-contained and tetherless (Boos et al., 2016), mobile HMDs can be used nearly anywhere, including outdoors and social gatherings (Pohl and de Tejada Quemada, 2016). Second, these systems are relatively affordable for the general population, as they only require a smartphone and a head-mounted peripheral (Gargantini et al., 2015). In turn, the low cost of these commercial devices make them broadly accessible (Steed et al., 2016). Due to these advantages, researchers have investigated mobile HMDs for treating amblyopia (Gargantini et al., 2015), self-managing pain (Tong et al., 2015), and geometry education (Steed et al., 2016).

However, mobile HMDs are severely limited for running VR applications compared to traditional VR systems. One of the biggest limitations for mobile HMDs is their limited graphics processing units (GPUs), which are over an order of magnitude slower than desktop-based GPUs (Boos et al., 2016). These mobile GPUs cannot handle the same virtual environments and objects as desktop-based VR systems without rendering at lower framerates, which can

[1] ©2018 ACM. Reprinted, with permission, from Kanchan Bahirat, Chengyuan Lai, Ryan P. Mcmahan, and Balakrishnan Prabhakaran. 2018. Designing and Evaluating a Mesh Simplification Algorithm for Virtual Reality. ACM Trans. Multimedia Comput. Commun. Appl. 14, 3s, Article 63 (June 2018), 26 pages. DOI: https://doi.org/10.1145/3209661

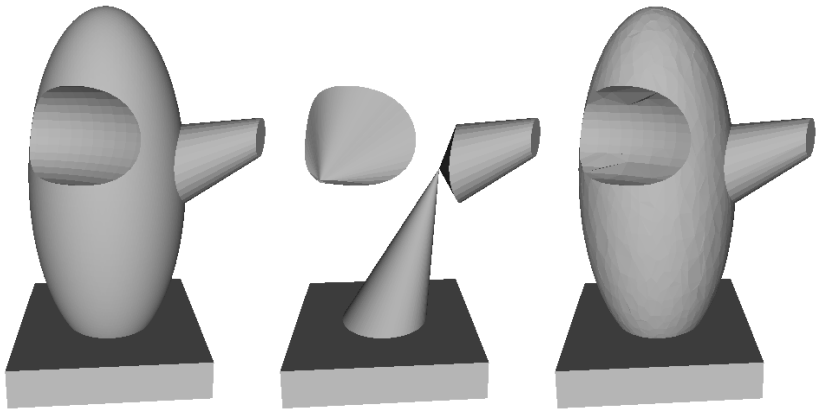
induce simulator sickness (Zielinski et al., 2016). Additionally, mobile HMDs are also constrained with little processing power and onboard memory leading to an upper bound on data size. For example, the Samsung Gear VR has a limit of 50K-100K polygons (Pruett, 2015) and the Microsoft HoloLens has a limit of 900 MB for memory (Microsoft, 2015).

There are approaches for addressing the GPU limitations of mobile HMDs. Boos et al. (Boos et al., 2016) have presented the FlashBack system, which precomputes and caches all possible images that a VR user might encounter instead of relying on real-time graphics rendering. In an evaluation, Boos et al. (Boos et al., 2016) found that FlashBack delivered better frame rates than a desktop-based VR system for both static and dynamic virtual environments. However, they acknowledged that the FlashBack system cannot handle several currently-visible dynamic objects or interactive lightning models, which are important requirements for many VR applications.

Another approach to mitigating the GPU limitations of mobile HMDs is to reduce the overall number of polygons that the GPU must render in real-time. One method to accomplish this is to artistically recreate 3D models by designing low-poly mesh versions. However, this method is extremely intensive in terms of an artist's time commitments. Alternatively, mesh simplification algorithms can be used to rapidly create low-poly versions of pre-existing, high-poly 3D models. However, many existing mesh simplification algorithms are not able to adequately handle manifold meshes with boundaries and non-manifold meshes. Many models, especially those made using computer-aided design (CAD) tools, are non-manifold meshes with boundaries (Luebke, 2001).

As most of the VR applications utilize CAD models, key requirements for designing a mesh simplification algorithm suitable for VR include the ability to handle: (a) very high-poly, non-manifold meshes; (b) preserve mesh boundaries; and (c) preserve mesh properties such as texture.

Most mesh simplification algorithms require a manifold mesh (Guéziec et al., 1998) as input (Luebke, 2001). As a result, only a few mesh simplification algorithms can handle most



(a) Original mesh (20,544 faces) (b) QEM_{BP} (3,236 faces) (c) QEM_{4VR} (3,084 faces)

Figure 2.1: Example of the QEM_{BP} approach's inability to preserve important surface edges.

3D models without failing. Of the simplification algorithms that can handle non-manifold meshes, the quadric error metric (QEM) approach is one of the most popular due to its speed and accuracy (Hoppe, 1999). However, because the original algorithm (Garland and Heckbert, 1997) allows for the collapse of boundary edges, it has the tendency to create large gaps at boundaries (see Figure 2.3(b)). To address this, Garland and Heckbert (Garland and Heckbert, 1998) provided a QEM modification for boundary preservation (QEM_{BP}) by weighting the boundaries when deciding which edges to keep. Unfortunately, this algorithm tends to preserve unnecessary boundary edges and collapsing needed surface edges, resulting in degraded surface approximations (see Figure 2.1(b)).

To address the requirements of VR applications and mobile HMDs, we have developed a novel mesh simplification algorithm utilizing the QEM infrastructure that eliminates the deficiencies of QEM and QEM_{BP}. This new algorithm, called QEM_{4VR}, uses a curvature-based boundary preservation approach to maintain key boundary edges while not sacrificing necessary surface edges. This curvature-based approach avoids creating gaps at boundaries and holes within surfaces (see Figure 2.6(d)). In addition to yielding more-accurate, low-poly meshes, it also preserves key surface properties, such as normals and materials (see

Figure 2.9(f)). In this paper, we present the details of our QEM_{4VR} mesh simplification algorithm that pre-process 3D models and generate their low-poly version offline. We also describe the conceptual differences between our approach and prior QEM variations. Additionally, we have used six publicly available, high-poly models with and without textures to compare the accuracy and fidelity of our QEM_{4VR} algorithm to previous QEM variations. Our results indicate that as meshes are simplified to approximately 10% of their original polygon totals, QEM_{4VR} maintains a higher degree of accuracy with fewer errors than the other algorithms. Additionally, the low-poly meshes generated by QEM_{4VR} have high-fidelity appearances compared to the original high-poly models, especially those with textures. Furthermore, the experimental results demonstrate that with QEM_{4VR}, we can achieve the frame rate of approximately 60 frames/second while retaining the high visual quality. The time complexity of the QEM_{4VR} is O(nlogn) which is same as original QEM. QEM_{4VR} also provides a scalability to simplify meshes with a polygon count of magnitude greater than 1000K polygons. Hence, QEM_{4VR} is a suitable algorithm for creating low-poly meshes for most VR applications, including mobile HMDs.

2.2 Related Work

Numerous mesh simplification algorithms have been previously proposed (see Cignoni et al. (Cignoni et al., 1998) and Luebke (Luebke, 2001) for two excellent surveys). However, a key requirement for designing an algorithm for VR is that the mesh simplification approach must be capable of handling non-manifold meshes, which are typical for models created by hand using CAD tools (Luebke, 2001). Unfortunately, most of the previous mesh simplification algorithms can only handle manifold meshes and fail on non-manifold meshes. Hence, in this section, we distinguish between manifold-only algorithms and non-manifold capable algorithms. Only the non-manifold capable algorithms are broadly applicable for developing VR applications.

2.2.1 Manifold-Only Simplification Algorithm

One approach to simplifying manifold meshes is to optimize the original one. Hoppe et al. (Hoppe et al., 1993) presented a mesh optimization algorithm that produces a new mesh of the same topological type as the original mesh, but with a smaller number of vertices. To produce new meshes, Hoppe et al. (Hoppe et al., 1993) defined three types of mesh transformations to apply to the original mesh: edge collapse, edge split, and edge swap. Hoppe et al. (Hoppe et al., 1993) randomly uses these transformations to produce new candidate manifold meshes, and then picks the candidate that minimizes an energy function that represents the error of the mesh.

Hoppe (Hoppe, 1996) later defined the concept of a progressive mesh, which represents a manifold mesh as a sequence of edge collapses. The original mesh can be retained by applying a series of vertex splits to the simplified mesh. Hoppe (Hoppe, 1996) also defined a new energy function to use during simplification to better represent mesh complexity and maintain a higher degree of fidelity.

Another approach to simplifying a manifold mesh is surface re-tiling, which involves defining a new set of vertices projected onto the surface of the original mesh and then using those vertices to form new surface faces. The surface re-tiling algorithm presented by Turk (Turk, 1992) first projects a new set of vertices onto the surface of the mesh, usually in a uniform pattern and spacing, though the curvature of a surface could be taken into account. The fidelity of this approach is highly dependent upon the method chosen to project the new set of vertices upon the surface.

The voxel-based simplification algorithm presented by He et al. (He et al., 1995) first segments a mesh into a voxel-based grid, and then applies a low-pass filter to eliminate voxels with the least over-lapping volumes with the original mesh. The algorithm then uses the marching cubes algorithm (Lorensen and Cline, 1987) to generate a new mesh based on

the remaining voxels. Due to its volume-based filtering approach, voxel-based simplification performs poorly on models with sharp edges and squared corners (Luebke, 2001).

Cohen et al. (Cohen et al., 1996) presented another approach to simplifying manifold meshes by using two offset copies of every mesh surface, called simplification envelopes. These offsets were used to ensure that a simplified surface would remain within the volume formed by the envelopes. While simplification envelopes can guarantee a high degree of fidelity given the offset distance used for the envelopes, the distance also serves as a lower bound for the algorithm and prevents drastic mesh simplification.

Yet another approach to simplifying manifold meshes is to use multiresolution analysis to decompose a function, representing the original high-poly mesh, into a simpler function, representing a low-poly mesh (Lounsbery et al., 1997). Eck et al. (Eck et al., 1995) presented an adaptive subdivision algorithm using multiresolution analysis to simplify arbitrary manifold meshes. The key contribution of their work was the design of a continuous parametrization of an arbitrary mesh over a simple domain mesh. The fidelity of the algorithm presented by Eck et al. (Eck et al., 1995) is relatively high for smooth organic forms (Luebke, 2001), but it fails to capture sharp features unless those features correlate to divisions in the base mesh (Hoppe, 1996). Hosseini et al. (Hosseini et al., 2012) presented an adaptive 3D texture streaming approach for M3G based mobile games. As this method aims at solely reducing transmission latency, it may not handle complex models for VR.

In summary, there have been many approaches to simplifying manifold meshes. Each approach has its strengths and weaknesses, in terms of the fidelity of the simplified meshes, the ability to define the degree of error, time performance, and the complexity of their implementations. However, these algorithms do not handle non-manifolds, and therefore, are inadequate for most VR applications.

2.2.2 Non-Manifold Simplification Algorithms

One of the first mesh simplification algorithms capable of handling non-manifold meshes was the decimation algorithm presented by Schroeder et al. (Schroeder et al., 1992). The decimation algorithm operates by making multiple passes over all the vertices of the model. During each pass, for each vertex, the algorithm determines whether a vertex can be removed without violating the topology of the local neighborhood of faces. If the resulting surface would be within a user-defined distance of the original mesh, the algorithm removes the vertex and all of its associated faces. The resulting hole is then re-triangulated. The decimation algorithm presented is capable of handling non-manifold meshes, if it does not delete non-manifold vertices during its removal passes (Luebke, 2001). However, this rule defines a lower bound for the decimation algorithm, as it cannot produce a low-poly mesh with fewer vertices than the number of non-manifold vertices. Additionally, the user-defined distance has a major impact on both the lower bound and fidelity of the algorithm (Garland and Heckbert, 1997).

Another approach to simplifying non-manifold meshes is vertex clustering. The vertex clustering algorithm presented by Rossignac and Borrel (Rossignac and Borrel, 1993) assigns an importance value to each vertex based on the size of its adjacent faces and its curvature. The algorithm then uses a 3D grid to segment the mesh and collapses all vertices within any given grid cell to the single most important vertex within the cell. The resolution of the grid determines the fidelity of the resulting simplified mesh. However, because the approach does not guarantee the amount of error introduced by the simplification, vertex clustering algorithms are often visually less pleasing than other mesh simplification algorithms (Luebke, 2001).

Another simplification approach that relies on vertex clustering is the hierarchical dynamic simplification (HDS) algorithm by Luebke and Erikson (Luebke and Erikson, 1997).

The HDS algorithm represents the entire mesh as a vertex tree, which is a hierarchy consisting of vertex clusters. The nodes of the tree can either be folded into their parent nodes to reduce the overall number of faces, or unfolded with their children nodes to re-obtain the original mesh. Because the vertex tree does not require vertex connectivity, the HDS algorithm supports non-manifold meshes. However, its fidelity tends to be less than the fidelity of other simplification algorithms (Luebke, 2001).

Perhaps the most common approach to simplifying non-manifold meshes is to use a quadric error metric (QEM), which was first presented by Garland and Heckbert (Garland and Heckbert, 1997). The QEM of a vertex is a 4 x 4 matrix that represents the sum of the squared distances from the vertex to the planes of adjacent faces. When a vertex is merged with another vertex within a user-defined distance threshold, the error introduced can be computed as the sum of the QEMs of the vertices being merged, which becomes the QEM of the new vertex. When merging vertices, the QEM algorithm keeps a sorted priority queue of all candidate vertex pairs based on their merged QEM. The algorithm removes the vertex pair with the lowest error from the top of the queue, merges the vertices, and then updates the errors of all vertex pairs involving the merged vertex. By using the QEM, this approach yields simplified meshes that are relatively high fidelity, even at drastic levels of simplification (Luebke, 2001).

One of the limitations of the original QEM algorithm is that it has the tendency to produce large deviations at boundary edges due to the decreased number of adjacent faces (Garland and Heckbert, 1997). To address this issue, and afford boundary preservation, Garland and Heckbert (Garland and Heckbert, 1997) also defined a boundary constraint plane passing through each boundary edge. For each boundary constraint plane, they computed a QEM that is multiplied by a high constant weight factor and added it to the initial QEMs of the edge vertices. Various researchers (Lindstrom and Turk, 1998; Garland, 1999; Wu et al., 2001) have also suggested weighing the boundary constraint plane by the square of the edge

length. However, all of these approaches result in prioritizing all boundary edges more than surface edges, which results in simplified meshes with holes in their surfaces.

Another limitation of the original QEM algorithm is that it did not account for surface properties, such as normals, colors, and texture coordinates. Garland and Heckbert (Garland and Heckbert, 1998) generalized the original QEM to also handle surface properties. To handle normals or colors, the original 4 x 4 error matrix can be extended to a 6 x 6 matrix that contains the error of the vertex's normal (abc) or its color (rgb). Similarly, a 5 x 5 error matrix can be used to capture the error of the vertex's texture coordinates (st). Hoppe (Hoppe, 1999) also presented a generalized QEM that required less storage space than that of Garland and Heckbert (Garland and Heckbert, 1998) by using a wedge-based mesh data structure. Along with his generalized QEM, Hoppe (Hoppe, 1999) also proposed a memoryless simplification algorithm and a volume-preserving algorithm. More recently, Ovreiu (Ovreiu, 2012) presented two quadric error metrics capable of handling surface attributes.

Since the introduction of QEM, many researchers have explored how to use it for various applications. For example, Pojar and Schmalstieg (Pojar and Schmalstieg, 2003) developed a plugin tool for Autodesk Maya that allows the user to create multiresolution meshes, including from non-manifold meshes. However, most variations of QEM are suitable for specific types of meshes (e.g., no boundaries, lots of boundaries, textured, etc.), but produce less-than-desirable results when used for other types of meshes. Because VR applications are so diverse, and their 3D models can vary drastically (e.g., manifold with no boundaries, manifold with boundaries, non-manifold), a single mesh simplification approach was not available. We address this issue with the presentation of our QEM_{4VR} algorithm.

2.2.3 Quality of Experience Assessment

With the ever-increasing number of interactive multimedia systems, researchers have become more concerned about the Quality of Experience (QoE) afforded by those systems. The

International Telecommunication Union has defined QoE as the "overall acceptability of an application or service, as perceived subjectively by the end-user" (Ebrahimi, 2009). However, as Timmerer et al. (Timmerer et al., 2015) have pointed out, a user might "accept" a multimedia system "without necessarily being happy or satisfied with it." Hence, we consider the definition of QoE proposed by Raake and Egger (Raake and Egger, 2014) to be more appropriate: "QoE is the degree of delight or annoyance of a person whose experiencing involves an application, service, or system. It results from the person's evaluation of the fulfillment of his or her expectations and needs with respect to the utility and/or enjoyment in the light of the person's context, personality and current state."

Researchers have investigated various methods to assess overall QoE for immersive systems, such as VR. Many of these methods involve taking both objective and subjective measurements to quantify various aspects of the user's experience. Puig et al. (Puig et al., 2012) proposed a QoE method primarily focused on objective user performance metrics, particularly task completion times. They conducted a pilot study using the proposed method and found that users became proficient at an object docking task much faster with a large projection screen than a computer screen. Within the VR community, there have been many studies that have investigated the effects of various interactive systems on user performance metrics. For example, McMahan et al. (McMahan et al., 2012) evaluated the independent effects of and interactions between display fidelity and interaction fidelity on user performance in a first-person shooter VR game. A review of similar user-performance studies is provided by Bowman and McMahan (Bowman and McMahan, 2007).

In addition to task performance, QoE researchers have investigated the use of heart rate (HR) and electrodermal activity (EDA) as objective QoE measures. In an early study, Meehan et al. (Meehan et al., 2002) found that HR significantly correlated to self-reported measures of presence (i.e., the sense of "being there") in an immersive VR system simulating walking above a pit, while EDA and skin temperature did not. More recently, Egan et al.

(Egan et al., 2016) found that EDA was significantly higher for viewing a city landscape virtual environment with a conventional 2D monitor as opposed to an HMD, and that most subjective QoE responses were better for the HMD condition than the 2D monitor condition. Potential reasons for the different EDA effects found by Meehan et al. (Meehan et al., 2002) compared to Egan et al. (Egan et al., 2016) are the improvement of EDA measurement devices and the contrasting nature of their virtual environments (i.e., acrophobia-oriented pit vs. calm city landscape). In other work, Keighrey et al. (Keighrey et al., 2017) used HR and EDA to compare a VR HMD to an Augmented Reality (AR) HMD for a speech and language assessment application. They found that the VR HMD elicited greater levels of HR and EDA, which indicate that the VR HMD yielded greater levels of arousal than the AR HMD.

Another approach to assessing QoE is to investigate the Quality of Sensory Experience (QoSE) (Timmerer et al., 2015). According to Raake and Egger (Raake and Egger, 2014), perceiving sensory stimuli is the basis of quality and QoE. Hence, many researchers have sought to establish Just Noticeable Differences (JNDs) for various sensory stimuli. A JND is the minimum amount that a stimulus must be changed for the difference to be noticeable to humans (Lecuyer et al., 2000). Many JND studies have focused on the visual qualities of multimedia. Yang et al. (Yang et al., 2005) established a JND model for spatial masking factors in video encodings. Using a similar model, Zhang et al. (Zhang et al., 2008) were able to estimate JNDs for viewing images by summing the effects of the visual thresholds into sub-bands. Beyond 2D multimedia, Zhao et al. (Zhao et al., 2011) conducted an experiment to establish a binocular JND (BJND) model for viewing stereoscopic images. Using the BJND model, Hachicha et al. (Hachicha et al., 2013) created the objective Stereo Image Quality Assessment (SIQA). More recently, Wu et al. (Wu et al., 2011) conducted a JND study to identify a new critical quality factor for 3D tele-immersive video called Color-plus-Depth Level-of-Detail (CZLoD). In this paper, we discuss how the a JND methodology can be used to better assess the visual quality of simplified meshes.

2.3 New Simplification Algorithm

Considering the various properties of handmade 3D models used in VR applications, we have designed and developed a new mesh simplification algorithm that can effectively create a low-poly version of nearly any 3D model. Our algorithm builds upon the original QEM approach presented by Garland and Heckbert (Garland and Heckbert, 1997). Hence, we refer to it as QEM_{4VR}. Our algorithm overcomes the limitations of the original QEM algorithm and other prior variations by employing a curvature-based boundary preservation approach and considering a wide variety of surface properties, even within the same mesh, when assigning error metrics. In section 3.1, we present the basics of the original QEM algorithm for completeness and for those readers unfamiliar with the approach. In section 3.2, we discuss the boundary-handling limitations of prior algorithms and how our curvature-based approach overcomes those limitations to yield higher-fidelity, low-poly meshes. In section 3.3, we discuss the surface-handling limitations of previous variations and how our new simplification algorithm accounts for the wide range of surface properties that are common with handmade 3D models. Finally, in section 3.4, we present a summary of the QEM_{4VR} algorithm.

2.3.1 Basics of Quadric Error Metric

For completeness and those readers unfamiliar with the approach, we discuss the details of the original QEM algorithm (Garland and Heckbert, 1997) here.

Pre-cleaning of Mesh

Handmade 3D models may consist of unreferenced, redundant vertices and redundant faces that are generated during the modelling process. Unreferenced vertices may be created due to inappropriate deletion of faces; while redundant vertices and faces are often created due

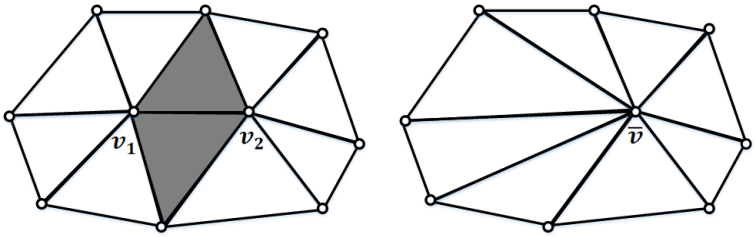


Figure 2.2: Example of collapsing an edge to a single vertex through the contraction $(v_1, v_2) \rightarrow \bar{v}$.

to specific file formats. Unreferenced, redundant vertices add unnecessary computational overhead and can also limit the ability to achieve drastic simplifications. Hence, when using any QEM approach, such redundant vertices and faces should be removed to accelerate processing and improve efficiency.

Mathematical Formulation

The original QEM method proposed by Garland and Heckbert (Garland and Heckbert, 1997) is based on the iterative contraction of vertex pairs. A pair contraction represented as $(v_1, v_2) \rightarrow \bar{v}$, moves vertices v_1 and v_2 to new position $\bar{v}$, connects all incident edges on v_1 and v_2 to $\bar{v}$, and deletes both v_1 and v_2, along with any degenerate edges and faces (see Figure 2.2).

At each step of the iteration, a valid vertex pair is chosen for contraction based on the cost of contraction measured by the geometric error of approximation. It characterizes the geometric error of an approximation using the quadric error metric. The QEM associates a set of planes with every vertex of the model. The geometric error induced by the removal of a particular vertex is defined to be the sum of the squared distance of the given vertex to all the planes in the associated set. Each set is initialized with faces incident to the given vertex in the original mesh. When the edge is contracted into a single vertex, the resulting set is the union of the two sets associated with endpoints of the edge. Every face in the original model is defined by a plane represented by the equation $n^T v + d = 0$, where $n = [n_x \; n_y \; n_z]^T$

is a unit normal and d is a constant. Given this representation of the plane, the squared distance of the vertex $v = [x\ y\ z]^T$ to the plane in given by (Garland and Heckbert, 1998):

$$D^2 = (n^T v + d)^2 = (v^T n + d)(n^T v + d) = v^T (nn^T)v + 2dn^T v + d^2 \tag{2.1}$$

Garland and Heckbert represent the D^2 using a convenient representation in terms of a *quadric* Q

$$Q = (A, b, c) = (nn^T, dn, d^2) \tag{2.2}$$

and corresponding quadric error analogous to D^2 is

$$Q(v) = v^T Av + 2b^T v + c \tag{2.3}$$

The addition of two quadrics can be computed componentwise: $Q_1(v) + Q_2(v) = (Q_1 + Q_2)(v)$, where $(Q_1 + Q_2) = (A_1 + A_2,\ b_1 + b_2,\ c_1 + c_2)$. Hence, the total quadric error at any vertex can be computed by summing up the quadrics for each plane in the set associated with that vertex. Furthermore, when the edge (v_1, v_2) is collapsed, the resulting quadric is merely an addition of two quadrics given by: $Q = Q_1 + Q_2$ and cost of contraction $(v_1, v_2) \rightarrow \overline{v}$ is given by $Q(\overline{v}) = Q_1(\overline{v}) + Q_2(\overline{v})$. Hence, associated set of planes with each vertex are just conceptual with no need to maintain them explicitly.

The original quadric metric proposed in (Garland and Heckbert, 1997) is heavily influenced by the tessellation or the structure of mesh. As the shape of the surface is more important than the way it is tessellated, the quadric error should be independent of the structure of mesh. To achieve it, Garland suggested an area-weighted quadric [garlandthesis].

Vertex Placement

While contracting edge (v_1, v_2), the selection of target position $\overline{v}$ should be made. In (Garland and Heckbert, 1997), Garland and Heckbert suggested two primary choices based on required space efficiency and approximation quality as: 1) subset placement and 2) optimal placement.

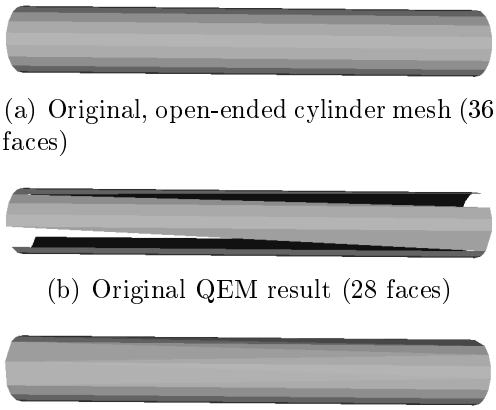
With the subset placement strategy, one of the end points is selected as a target position. The end point with smaller $Q(v)$ is contracted into another end point. This strategy produces an approximation using a subset of the original set of vertices with their original positions. Another possibility is to use the mid-point of the edge (v_1, v_2) as a target position.

With the optimal placement strategy, for a given quadric, the target position $\overline{v}$ is computed such that $Q(\overline{v})$ is minimal. As $Q(\overline{v})$ is quadratic, its minimum occurs at $\frac{\partial Q}{\partial x} = \frac{\partial Q}{\partial y} = \frac{\partial Q}{\partial z} = 0$. By solving this linear system, we can find the optimal target position and corresponding error as: $\overline{v} = -A^{-1}b$ and $Q(\overline{v}) = -b^T A^{-1} b + c$. If the matrix A is not invertible, we can use a subset placement.

Optimal placement results in the better approximation closely fitting to the original mesh. Resultant meshes have more equilateral triangles with uniform areas. On the other hand, subset placement results in inferior approximation, but needs significantly less storage space. With both optimal placement and mid-point placement, one needs to store all delta changes for every contraction. This overhead can be avoided in subset placement. Hence, we have chosen to use the subset placement policy in our QEM_{4VR} method. However, QEM_{4VR} can be easily adapted to any of the above-mentioned policies.

2.4 Curvature-Based Boundary Preservation

The key insight that our QEM_{4VR} algorithm leverages is the fact that boundary edges are important for the perception of the shape of a mesh, but that not every boundary edge needs to be kept to maintain that shape. In particular, we have used the curvature of a boundary edge vertex to judge the importance of its boundary edges. Thus, our algorithm better maintains both the boundaries and the surface edges of a mesh during simplification, which results in higher-fidelity simplified meshes.



(a) Original, open-ended cylinder mesh (36 faces)

(b) Original QEM result (28 faces)

(c) QEM with boundary constraints result (28 faces)

Figure 2.3: Example of the original QEM algorithm's inability to preserve important boundary edges.

2.4.1 Limitations of Original QEM Approach

Generally, sharp edges on the surface of a mesh and its boundaries characterize the object and are considered the most significant visual features (Luebke, 2001). Therefore, it becomes critical to preserve these features for obtaining a better sparse approximation. The original QEM algorithm (Garland and Heckbert, 1997) inherently handles surface shape discontinuities and preserves sharp edges. For example, consider a sharp edge of a cube. For the vertices on the edges of the cube, contributing faces come from the adjacent faces of the cube, which are perpendicular to each other. Hence, the cost of moving the vertex along the edge will be less than moving the vertex away from the edge. Therefore, the original QEM algorithm is strongly biased against altering the sharp features of a mesh's surface. On the other hand, the original QEM algorithm does not account for actual mesh boundaries (i.e., edges with only one adjacent face). Instead, it has the tendency of prioritizing sharp surface edges over boundary edges. As a result, those boundary edges are removed, which in turn, causes holes to form within the surfaces that the algorithm was trying to preserve (see Figure 2.3(b)).

2.4.2 Limitations of Prior Boundary Constraints

Recognizing the original algorithm's inability to preserve important boundary edges, Garland and Heckbert (Garland and Heckbert, 1997) suggested using a boundary preservation approach by marking the boundary edges during the initialization process. For each boundary edge, they computed a plane perpendicular to the edge's face that passed through the edge. Similar to the regular plane, a quadric is computed for these boundary constraint planes. A constant and large weight factor is applied to resultant quadric and added to the initial quadrics for both of the boundary edge's vertices. For the area-weighted quadric, they applied a squared length of the boundary edge as the weight factor. Figure 2.3 shows the results before and after applying the boundary constraints described.

However, this boundary preservation approach is not without its own limitations. The approach is highly biased towards not altering the boundary vertices and may result in inferior results. It will remove most surface edges and their vertices before removing any boundary vertices. For example, consider the mesh drastically simplified with boundary constraints in Figure 2.1.

2.4.3 Curvature-Based Boundary Preservation Approach

To achieve a better boundary constraint, it is important to consider the shape of the boundary curve. For example, the vertices on a linear boundary may be removed in order to keep some of the vertices on the inner surface to get a better approximation. Because the existing boundary preservation approach applies a constant weight factor, it removes the vertices from the inner surface instead of vertices on the linear boundary. Hence, it results in a degraded mesh. Here, we describe our curvature-based boundary preservation approach.

We first identify the boundary edges during initialization. For manifold and non-manifold surfaces, if the edge is shared by two or more triangles, it is considered interior. If the edge

is shared by only one triangle, it is considered as a boundary edge and the corresponding endpoints are considered as boundary vertices.

As the proposed approach is designed for both manifold and non-manifold meshes, boundary preservation should also handle both types of meshes. First, we discuss the case of a manifold mesh with boundaries. Every boundary vertex will have exactly two neighboring boundary vertices. Consider the boundary vertex v_2 and v_3. We compute the curvature of the boundary curve at vertex v_1 as (Goldman, 2005):

$$\begin{aligned} x' &= v_3.x - v_2.x, \quad & x'' &= v_3.x - 2 * v_1.x + v_2.x, \\ y' &= v_3.y - v_2.y, \quad & y'' &= v_3.y - 2 * v_1.y + v_2.y, \\ z' &= v_3.z - v_2.z, \quad & z'' &= v_3.z - 2 * v_1.z + v_2.z. \end{aligned} \tag{2.4}$$

$$k = \frac{\sqrt{(z''y' - y''z')^2 + (x''z' - z''x')^2 + (y''x' - x''y')^2}}{(x'^2 + y'^2 + z'^2)^{\frac{3}{2}}} \tag{2.5}$$

where k is a curvature of boundary curve at v_1, x', y' and z' are first order derivatives and x'', y'' and z'' are second order derivatives of underlying surface in respective directions.

For non-manifold surfaces, the assumption of exactly two neighboring boundary vertices may not hold true. Consequently, the curvature computed considering any random pair of neighboring boundary vertices will result in an incorrect judgment about the vertex. To handle such scenarios, we simply mark non-manifold boundary vertices as complex vertices and do not simplify them by assigning them high weights. Inspired by the original QEM mesh simplification method (Garland and Heckbert, 1997), we compute a boundary constraint plane and corresponding quadric in a similar manner. Computation of a quadric for boundary constraint planes allows incorporating boundary preservation without altering the functionality of the original algorithm. Next, we add the boundary constraint plane's quadric to both endpoints of the boundary edge. Instead of multiplying it with the constant high weight factor, we multiply the quadric with the curvature at each endpoint independently,

and then add it to initial quadrics of the respective vertices. Mathematically,

$$\begin{aligned} Q(v_1) &= Q(v_1) + W_b * k(v_1) * Q_{bcp} \\ Q(v_2) &= Q(v_2) + W_b * k(v_2) * Q_{bcp} \end{aligned} \tag{2.6}$$

where, $Q(v_1)$, $Q(v_2)$ are quadrics associated with the vertex v_1 and v_2, Q_{bcp} is a quadric of boundary constraint plane at edge (v_1, v_2), W_b is the user defined weight factor and $k(v_1)$ and $k(v_2)$ are curvature of boundary curve at the vertex v_1 and v_2, respectively.

This curvature-based weighing scheme for preserving the boundary assigns a higher weight to boundary vertices with high curvature. The boundary vertices having less curvature (e.g., a vertex on a linear boundary) are assigned smaller weight. It will avoid highly biasing the algorithm toward boundary vertices. Our algorithm removes some of the vertices on the linear boundary to achieve better approximation across the surface edges. See Figures 2.5-2.10 for results of our curvature-based approach.

2.4.4 Preserving Multiple Surface Properties

Another key contribution of our QEM_{4VR} algorithm is its ability to handle multiple surface properties. Prior QEM variations that preserve surface properties have been presented (Hoppe, 1999; Garland and Heckbert, 1998). However, these approaches make several assumptions regarding surface properties that are not typical for handmade 3D models. In turn, these prior preservation approaches often fail for VR applications. To overcome these limitations, we have identified key cases that are common to handmade 3D models and must be handled to preserve the surface properties of a mesh.

Limitations of Prior Surface Property Approaches

Nearly all 3D models used in VR applications possess surface properties, in addition to their geometric properties. The most common properties are normals, texture coordinates, colors,

and materials. The associated texture information has a high impact on the perceptual quality. Hence, to obtain an approximation with high visual fidelity, it is crucial to maintain these surface properties.

In their later work, Garland and Heckbert (Garland and Heckbert, 1998) proposed a generalized error metric that is an extension of the basic quadric error metric. This generalized metric incorporated the surface properties as vertex attributes to provide surface property-based simplification. They treated each vertex as a vector $v \in R^n$. Where, first three components of v are spatial coordinates and remaining components represent property values. Hence, for any vertex with associated three color components, we have $v = [x\ y\ z\ r\ g\ b]$ and $n = 6$. For n-dimensional vertices, the modified squared distance D^2 of v from any triangle T also has the structure of original quadric metric (Garland and Heckbert, 1998) as: $D^2 = v^T Av + 2b^T v + c$ where,

$$\begin{aligned} A &= I - e_1 {e_1}^T - e_2 (e_2)^T \\ b &= (p.e_1)e_1 + (p.e_2)e_2 - p \\ c &= p.p - (p.e_1)^2 - (p.e_2)^2 \\ e_1 &= \frac{q-p}{||q-p||} \\ e_2 &= \frac{r - p - (e_1.(r-p))e_1}{||r - p - (e_1.(r-p))e_1||} \end{aligned} \tag{2.7}$$

In this generalized quadric metric, (p, q, r) are vertices of triangle T, A is $n \times n$ matrix and b is $n \times 1$ vector. One can easily replace color values with texture coordinates to create a n dimensional vector for each vertex. This approach is suitable if the entire model is mapped to a single texture material such that there is a one-to-one mapping between vertices and texture coordinates. But in practice, multiple texture materials can be associated with a single 3D model. Furthermore, multiple values from a single texture map may also be assigned to the given vertex. Hence, a model may consist of vertices with multiple values of the associated texture coordinates. Hoppe (Hoppe, 1999) also suggested a QEM to handle

surface attributes, but both surface property preservation methods assume that every vertex will have at least one surface property defined, in addition to its geometric property. However, that is not the case for many handmade 3D models.

Multiple Surface Properties Preservation Approach

To better handle and preserve the surface properties of handmade 3D meshes, we first must identify the common cases that should be expected by our simplification algorithm. We have identified the following common cases involving surface properties:

- A vertex may not have any surface properties.
- Multiple textures may be mapped to the mesh.
- A vertex may be associated with multiple coordinates within the same or different texture maps.

To handle these cases, we have created a new data structure for each vertex that maintains its neighboring faces and corresponding texture coordinates associated with the given vertex. We update these data structures during edge collapse with sub-optimal vertex placement for adequate attribute transfer. We have also extended our boundary preservation approach to preserve the edges that form boundaries between multiple textures. The edges that are associated with multiple textures constitute material boundaries. This novel approach consists of:

- Defaulting texture coordinates and material indices to zero for faces with no surface properties.
- Identifying vertices that form material boundaries.
- Identifying critical vertices that are associated with multiple materials or textures.

- Assigning the user-defined weight W_t to the metrics of material boundary vertices and critical vertices.

- Transferring the attributes during each edge collapse.

If the model has a subset of faces that do not have surface property information, we set corresponding properties such as texture coordinates and material indices to zero. Because the vertices of these faces do not have surface properties, setting the surface property values to zero will make sure that the order of associated edge collapses is based solely on the geometric error and not surface properties that do not exist. Next, to identify the material boundaries, we utilize a structure that maintains the list of faces in the neighborhood of a vertex. If any two faces in this list have a different material index, we can consider that the given vertex has been assigned two different materials. Hence, it can be identified as a material boundary vertex. Further, if any vertex is associated with multiple texture values, then it is considered a critical vertex. Next, we apply a user-defined, weight factor W_t to the quadric metrics associated with the vertices on the material boundary. This way, we can make sure that the material boundaries remain unaltered and material indices are properly preserved. Because we have adopted a subset placement policy, handling the texture coordinates during the edge collapse is a trivial task for models with one-to-one mapping between vertices and texture coordinates. But for models with multiple attributes per vertex, extra care is needed. Consider the edge collapse (v_1, v_2), as shown in Figure 2.4. Neighboring faces with the same color indicate that a vertex has the same texture coordinate value for all of the faces. Hence, vertex v_1 has two texture coordinate values associated with it; one texture coordinate is common for faces 1, 2, and 3, while another texture coordinate is shared by faces 4, 5, and 6. As we use a sub-optimal vertex placement, the vertex v_1 is moved to vertex v_2 during edge collapse. Faces 3 and 4 are removed, while faces 1, 2, 5, and 6 form a set of affected faces. For the affected faces, the vertex v_1 is replaced by vertex v_2.

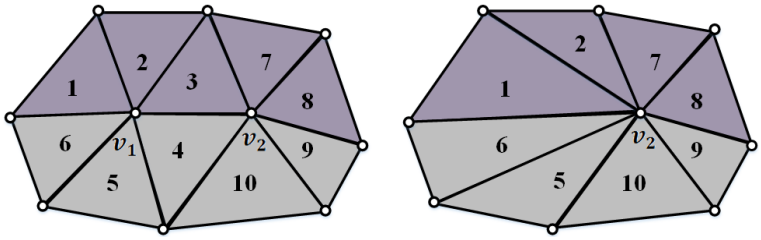


Figure 2.4: Attribute transfer while collapsing the edge (v_1, v_2) to vertex v_2 where each vertex has multiple attribute values.

Because both the vertices v_1 and v_2 have multiple texture coordinates, when we replace v_1 with v_2, we must update the texture coordinate associated with the affected faces at vertex v_1 with the valid texture coordinate at the vertex v_2. For example, when face 5 with the gray color is updated, we should modify its texture coordinate associated with v_1 with that of v_2 corresponding to gray color. To achieve this, we find the closest match for the texture coordinate of v_1 for the affected faces in the set of texture coordinates associated with the vertex v_2, with the additional constraint of matching material indices. Hence, for faces 1 and 2, the texture coordinate of v_1 is replaced with the texture coordinate of v_2 related to purple; but it is replaced with the texture coordinate of v_2 represented by the gray color for faces 5 and 6. This strategy allows achieving an adequate attribute transfer during edge collapses, as illustrated in Figure 2.4.

2.4.5 Summary of New QEM_{4VR} Approach

To summarize, the QEM_{4VR} algorithm employs the following steps to reduce any 3D model (manifold or non-manifold) into a low-poly mesh with approximately N (which is user-defined) faces that approximate the surface properties of the original mesh.

Step 1: Remove any redundant vertices and faces that were created during the handmade construction of the 3D model.

Step 2: For each vertex v, create the data structures T_v and $T_(x_v)$ representing the neighboring faces of vertex v, and its texture coordinates, respectively.

Step 3: For each vertex, determine if it is a complex vertex lying on the mesh boundary or not. If it lies on the boundary, determine the boundary curvature at the vertex using Equation 5. If it is a non-manifold vertex on the boundary, mark it as a complex vertex. Also, determine if the vertex is on a material boundary, or not utilizing its data structures T_v and $T_(x_v)$.

Step 4: Compute quadric metric for each vertex. For vertices on the mesh boundary, weigh the quadric metric by its corresponding curvature value and the user-defined weight factor W_b. Further, if the vertex lies on a material boundary, weigh the corresponding quadric with a user-defined weight factor W_t. Note that edges involving non-manifold boundary vertices (marked as complex vertex) are not collapsed.

Step 5: For each edge in the 3D model, compute a quadric cost that is the sum of the cost associated with its two endpoints.

Step 6: Select the edge with the minimum quadric error. To collapse the selected edge with vertices v_1 and v_2, we select the subset placement policy for determining the target position. If v_2 is selected as a target position, we merge the set of neighboring faces of v_1 with that of v_2, update the attributes for the merged faces using the described attribute transfer strategy. Next, we update the quadric error associated with v_2 and the list of edges.

Step 7: Repeat steps 2-7 until the poly count is less than N.

2.5 EXPERIMENTS

We have performed three sets of experiments to demonstrate the effectiveness and quality of our proposed QEM_{4VR} method.

- The first was designed to compare approximation quality when high-poly models are drastically simplified.
- In the second, we analyzed the error induced during progressive simplification at different levels of sparseness.

Table 2.1: Attributes of Datasets used for experiments.

	Mesh Type	**Faces**	**Vertices**	**Boundary Edges**
Non-Textured Models				
RockerArm	Manifold	80,354	40,177	0
Car	Manifold	268,630	202,890	120,227
Dragon	Non-manifold	37,986	22,126	6,214
Textured Models				
Head	Manifold	17,684	8,844	0
Jet	Manifold	35,095	18,849	2,341
Airplane	Non-manifold	66,496	36,937	6,686

- The last experiment consisted of frame rate analysis to study the latency gain obtained with low-poly models.

For the non-textured models, we compared our QEM_{4VR} algorithm to the original QEM algorithm (QEM_{ORIG}), which does not preserve boundaries, and the QEM algorithm with boundary constraints (QEM_{BP}). For the textured models, we compared our QEM_{4VR} algorithm to a texture preservation QEM variation (QEM_{TP}) and a boundary and texture preservation QEM variation (QEM_{BTP}), in addition to QEM_{ORIG} and QEM_{BP}.

Setup. The QEM_{4VR} algorithm was implemented in C++, and the experiments were conducted on a computer with Intel®Core™i7-5820K with 3.30GHz speed and 32GB internal RAM. For other methods that are considered for comparison, we used their MeshLab [34, 35] implementations. We used OpenGL for rendering the resultant low-poly meshes. To handle 3D data, we have developed our own C++ data-structures and APIs.

Dataset. We evaluated the performance of our new QEM_{4VR} algorithm with a dataset consisting of a variety of 3D models. The dataset included manifold meshes without boundaries, manifold meshes with boundaries, and non-manifold meshes, all with and without textures. See Table 2.1 for a summary of the 3D models.

Parameters. User-defined weight factors W_b and W_t help to prioritize surface and material boundaries. Along with boundary curvature, above mentioned weight factors have

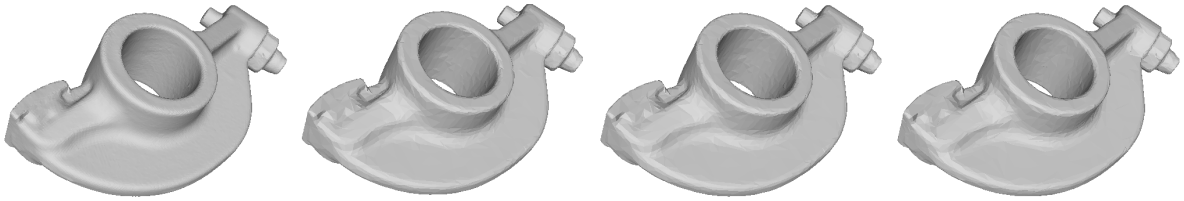
Table 2.2: Mean approximation errors (lower is better) for simplifying non-textured models to roughly 10% original size.

Non-Textured Models	**Number of Faces**				**Metro** 10^{-3}		
	Original	QEM_{ORIG}	QEM_{BP}	QEM_{4VR}	QEM_{ORIG}	QEM_{BP}	QEM_{4VR}
RockerArm	80,354	8,174	8,174	8,174	0.181	0.181	**0.180**
Car	268,630	24,176	$104,461^{*}$	24,258	1.245	1.018	**0.473**
Dragon	37,986	3,566	$6,156^{*}$	3,567	1.397	1.657	**0.341**

a significant impact on the order of edge collapse. Hence, these weights must be selected carefully. We experimented with various values of W_b and W_t ranging from 1 to 1000. For brevity, we have not included results with varying values of W_b and W_t. However, we found that W_b being 5 and W_t being 1000 are most suitable for all the models in the experimental dataset. For other variations of QEM, we set the quality threshold to the default 0.3 in MeshLab. For the boundary preservation variations, we set the boundary weight to 1 for all models, because lowering the boundary preserving weight does not have a drastic effect on the final output.

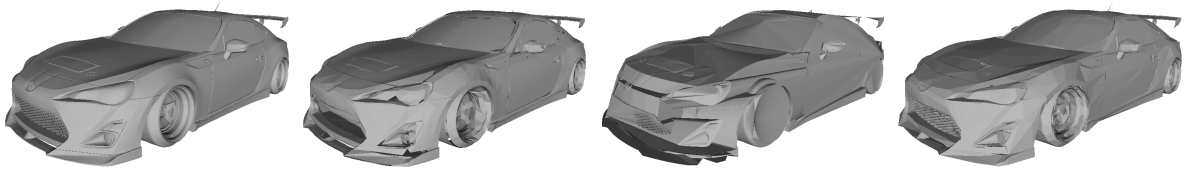
Metrics. To quantify how well the low-poly meshes generated by the various QEM approaches approximate the original meshes, we used an approach similar to prior researchers (Hoppe, 1999; Garland and Heckbert, 1997). We computed the distance between two meshes by sampling a set of points from the original high-poly mesh and then measuring their distance to their closest point on each low-poly mesh. This provides a one-sided approximation error. The average computed errors, called the mean approximation error or Metro (Cignoni et al., 1996), is normalized with respect to the diagonal length of the original model's bounding box and is used as the measurement for error.

To compare the quality of our new QEM_{4VR} algorithm to other QEM variations, we computed a texture error between the original model and each low-poly mesh as described in (Hoppe, 1999). The texture error is estimated by measuring the deviation of the texture coordinates of sampled points from values linearly interpolated at their projection on the closest face of the low-poly model.



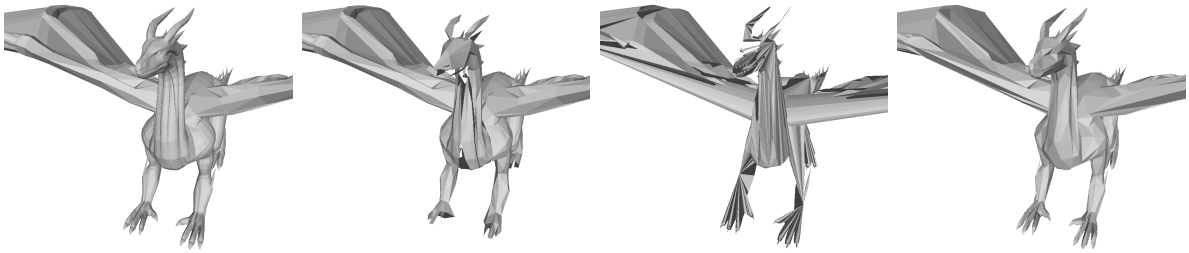
(a) Original model (b) QEM_{ORIG} (c) QEM_{BP} (d) QEM_{4VR}

Figure 2.5: Simplification of RockerArm (manifold with no boundaries) to approximately 10% its original number of faces.



(a) Original model (b) QEM_{ORIG} (c) QEM_{BP} (d) QEM_{4VR}

Figure 2.6: Simplification of Car (manifold with boundaries) to approximately 10% its original number of faces.



(a) Original model (b) QEM_{ORIG} (c) QEM_{BP} (d) QEM_{4VR}

Figure 2.7: Simplification of Dragon (non-manifold) to approximately 10% its original number of faces.

2.5.1 Drastic Simplification

In the first experiment, we simplified all the models to 10% of their original size using the proposed QEM_{4VR} and other variations of QEM. For each low-poly mesh, we computed Metro and texture error as described previously.

Geometric Approximations. Tables 2.2 and 2.3 provide the quantitative results by the different methods for errors introduced during simplification of the non-textured and textured models, respectively. It can be clearly seen that our new QEM_{4VR} algorithm results in the least amount of error and well approximates the original models. Figures 2.5-2.7 and 2.8-2.10 show the simplified non-textured and textured models, respectively.

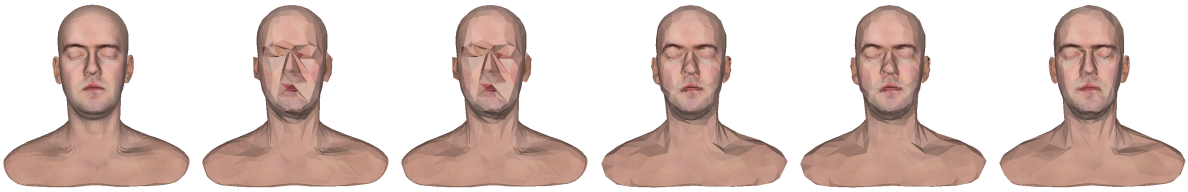
Preservation of Multiple Textures. Various results, as shown in Figures 2.8-2.10, demonstrate the ability of our QEM_{4VR} algorithm to preserve surface properties, including multiple textures. QEM_{4VR} efficiently handles the attribute transfer during simplification such that the resultant low-poly mesh well approximates the original model without introducing any visible degradation in the texture mapping. Table 2.4 provides the quantitative texture error measures for the different approaches. Our QEM_{4VR} method induces less texture error than the other QEM variations. Also, it is important to point out that both QEM_{TP} and QEM_{BTP} failed on the non-manifold Airplane mesh due to it containing vertices without associated texture coordinates.

Table 2.3: Mean approximation errors (lower is better) for simplifying textured models to roughly 10% original size.

Non-Textured Models	**Number of Faces**						**Metro** 10^{-3}				
	Original	QEM_{ORIG}	QEM_{BP}	QEM_{TP}	QEM_{BTP}	QEM_{4VR}	QEM_{ORIG}	QEM_{BP}	QEM_{TP}	QEM_{BTP}	QEM_{4VR}
Head	17,684	1,844	1,844	1,844	1,844	1,844	2.553	2.553	1.391	1.391	**0.911**
Jet	35,095	3,104	3,455	3,504	3,505	3,505	1.071	3.267	1.318	2.316	**0.340**
Airplane	66,496	6,796	6,796	$FAIL^*$	$FAIL^*$	6,797	0.941	3.415	$FAIL^*$	$FAIL^*$	**0.237**

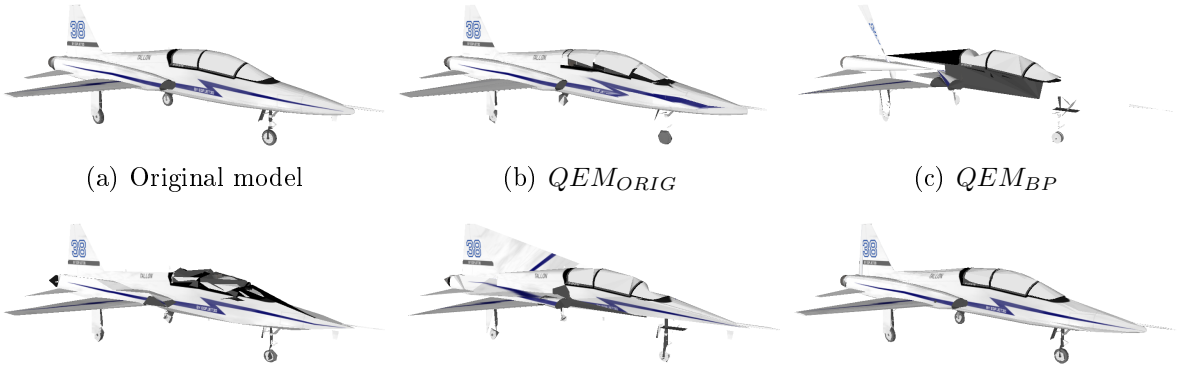
Table 2.4: Texture errors (lower is better) for simplifying textured models to approximately 10% original size.

Non-Textured Models	**Number of Faces**						**Texture Error** 10^{-2}				
	Original	QEM_{ORIG}	QEM_{BP}	QEM_{TP}	QEM_{BTP}	QEM_{4VR}	QEM_{ORIG}	QEM_{BP}	QEM_{TP}	QEM_{BTP}	QEM_{4VR}
Head	17,684	1,844	1,844	1,844	1,844	1,844	1.02	1.02	0.87	0.87	**0.34**
Jet	35,095	3,104	3,455	3,504	3,505	3,505	3.03	4.89	3.07	3.45	**2.01**
Airplane	66,496	6,796	6,796	$FAIL^*$	$FAIL^*$	6,797	7.19	8.13	$FAIL^*$	$FAIL^*$	**2.26**



(a) Original model (b) QEM_{ORIG} (c) QEM_{BP} (d) QEM_{TP} (e) QEM_{BTP} (f) QEM_{4VR}

Figure 2.8: Simplification of Head (textured manifold with no boundaries) to approximately 10% its original number of faces.



(a) Original model (b) QEM_{ORIG} (c) QEM_{BP}

(d) QEM_{TP} (e) QEM_{BTP} (f) QEM_{4VR}

Figure 2.9: Simplification of Jet (textured manifold with boundaries) to approximately 10% its original number of faces.

2.5.2 Progressive Simplification

To further evaluate the efficacy of the proposed method to generate the high fidelity sparse models, we obtained various low-poly models with different levels of sparseness (i.e., with a different number of faces) for Dragon and Jet model. For these low-poly models, we computed the mean approximation error or Metro. Figure 2.11 illustrates the behavior of Metro while reducing the number of faces for Dragon and Jet model. It can be seen that QEM_{4VR} introduces minimal error and consequently maintains high fidelity even at a very high level of sparseness. Further, QEM_{4VR} causes minimum approximation error across different levels of sparseness compared to the other QEM variations. It is also important to note that for both QEM_{BP} and QEM_{BTP}, the error curve terminates midway indicating

(a) Original model (b) QEM_{ORIG} (c) QEM_{BP} (d) QEM_{4VR}

Figure 2.10: Simplification of Airplane (textured non-manifold with boundaries) down to approximately 10% its original size. Note that QEM_{TP} and QEM_{BTP} failed to produce results due to their inability to handle vertices with multiple texture coordinates.

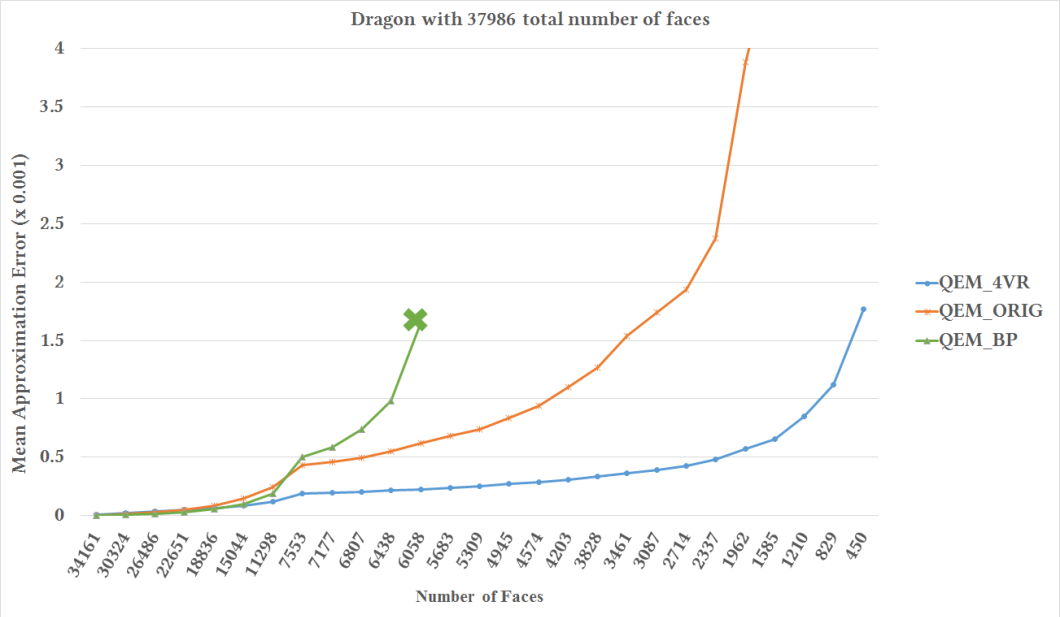
Table 2.5: Frame rate analysis.

# of Models	Original		QEM_{ORIG}		QEM_{BP}		QEM_{4VR}	
	Polygon Count	Frame Rate	Polygon Count	Frame Rate	Polygon Count	Frame Rate	Polygon Count	Frame Rate
1	66,496	57.61	6,796	57.68	6,796	57.76	6,797	57.78
2	146,850	57.39	14,970	57.62	14,970	57.62	14,971	57.65
3	181,945	52.18	18,074	57.59	18,425	57.35	18,476	57.65
4	450,575	37.71	42,250	57.50	122,886	50.26	42,734	57.63
5	468,259	37.36	44,094	57.49	124,730	50.25	44,578	57.55
6	506,245	35.33	47,660	57.49	130,886	47.92	48,145	57.49

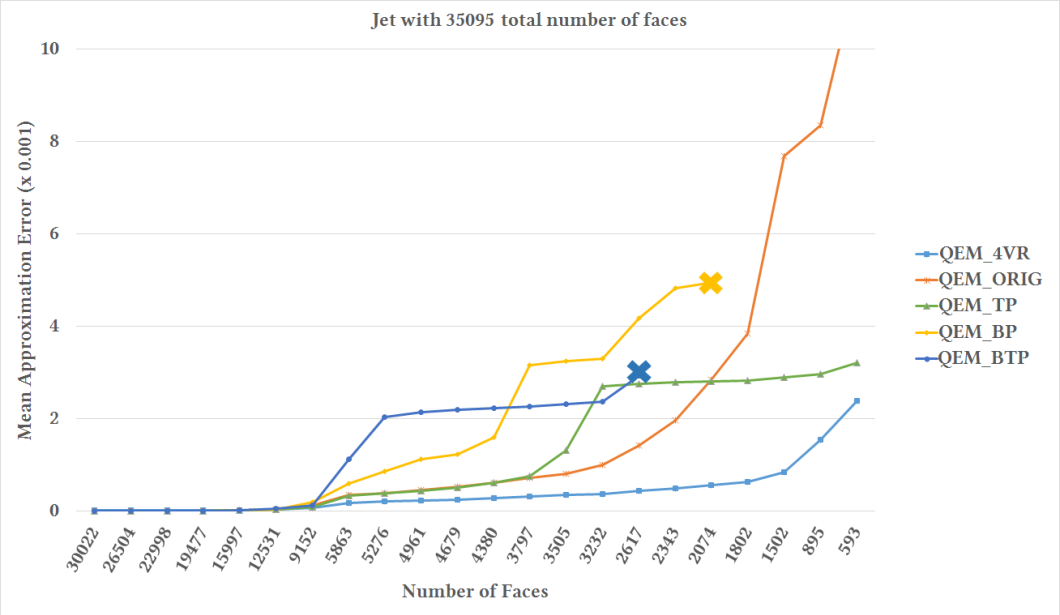
that no further poly count reduction can be achieved with the respective method due to the lower bound on simplification enforced by corresponding boundary constraints.

2.5.3 Frame Rate Analysis

To analyze the frame rate, we designed an experimental setup where four different scenes were generated using Unity3D game engine. The first scene was created utilizing the original six dense models from our dataset (see Figure 2.12). Those models were added into the scene and rendered in a sequential manner. We recorded the average frame rates every time a new model was added to the scene. Similarly, the other three scenes were created by using sparse models generated by QEM_{ORIG}, QEM_{BP}, QEM_{4VR}. Experiments were performed on Samsung Gear VR with Samsung S6 edge as a mobile device. Table 2.5 shows the frame rate performance while rendering the aforementioned scenes. It can be seen that as the



(a) For Dragon model



(b) For Jet model

Figure 2.11: Mean approximation error (Metro) with respect to the number of faces in simplified models.

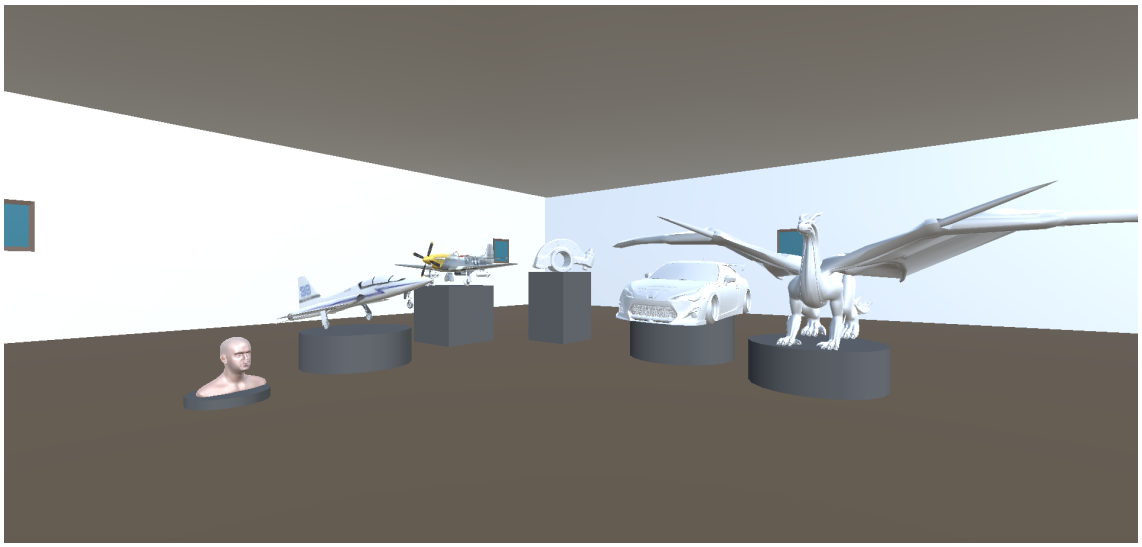


Figure 2.12: The scene consisting of low-poly models generated using QEM_{4VR} for the frame rate analysis experiment.

polygon count increases, the frame rate drops significantly in the case of the scene generated using dense models and sparse models obtained by QEM_{BP}. On the contrary, the scenes generated using sparse models obtained by QEM_{ORIG}, QEM_{4VR} maintain the average frame rate of approximately 58 frames per second.

2.5.4 Perceptual Quality Assessment

In VR applications, QoE is a crucial factor (Egan et al., 2016). Hence, it becomes essential to quantitatively measure the visual quality of our simplified meshes in correlation with the human perception. Various perceptual quality measures have been investigated. A mesh structural distortion measure (MSDM) proposed by Lavoué et al. (Lavoué et al., 2006) is one of the earlier works on objective perceptual mesh quality. Lavoué et al. (Lavoué et al., 2006) extended the well-known structural similarity index for 2D images (Wang et al., 2004) to 3D triangular meshes. MSDM utilizes the changes in the local statistics (i.e., mean, variance and covariance) of a surface curvature to quantify a visual degradation in the mesh quality. Later, Lavoué et al. (Lavoué, 2011) proposed an improved version (MSDM2) that

performs a vertex matching preprocessing step to allow the comparison of two meshes with different topologies.

Recently, Wang et al. (Wang et al., 2012) proposed the fast mesh perceptual distance (FMPD) metric. FMPD measures the local surface roughness derived from the Gaussian curvature of the mesh surface. It estimates the perceptual distance as the deviation of a global roughness of the modified mesh from a global roughness of the reference mesh. Hence, it does not require a mesh registration pre-processing step. Consequently, it can be applied to compare two meshes with different connectivity. Furthermore, FMPD does not need the full information about the original reference mesh, and can be considered as a reduced-reference metric.

Torkhani et al. (Torkhani et al., 2014) proposed a curvature tensor-based perceptual distance measure (TPDM) that utilizes tensor eigenvalues and eigenvectors to derive a perceptually oriented distance metric. TPDM incorporates roughness-based weighting of the local tensor distance in order to account for the visual masking effect of the human visual system. TPDM also performs a vertex matching as a pre-processing similar to MSDM2.

Among the available perceptual quality metrics, MSDM requires that the meshes to be compared share the same connectivity. Hence, it cannot be applied in simplification evaluation (Lavoué, 2011). Although, MSDM2 and TPDM perform vertex matching as a pre-processing step to avoid the same connectivity constraint, these methods do not handle non-manifold meshes. Hence, we utilized FMPD as a quantitative metric for assessing the perceptual quality of our simplified meshes.

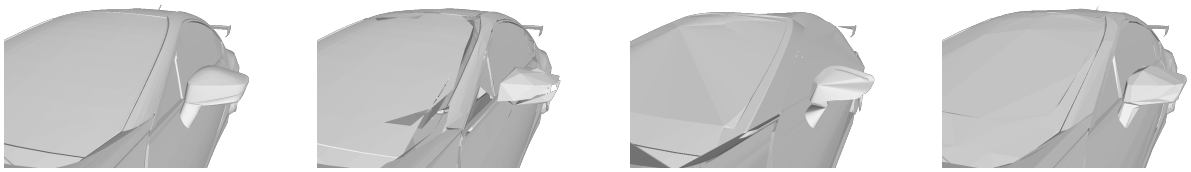
Experimental setup: To perform a perceptual quality assessment of QEM_{4VR}, we utilized the open source implementation of FMPD available at (Wang, 2012). We simplified all the models in Table 2.1 to 10% of their original size using the proposed QEM_{4VR} and other variations of QEM. For each low-poly mesh, we computed its FMPD (see Table 2.6 and 2.7). Note, FMPD values are not clipped to the range (0, 1).

Table 2.6: FMPD metric (lower is better) for simplifying non-textured models to roughly 10%.

Non-Textured Models	**FMPD**		
	$\boldsymbol{QEM_{ORIG}}$	$\boldsymbol{QEM_{BP}}$	$\boldsymbol{QEM_{4VR}}$
RockerArm	**0.408**	**0.408**	0.413
Car	**1.039**	1.921	1.744
Dragon	**0.185**	0.909	0.220

Table 2.7: FMPD metric (lower is better) for simplifying textured models to roughly 10%.

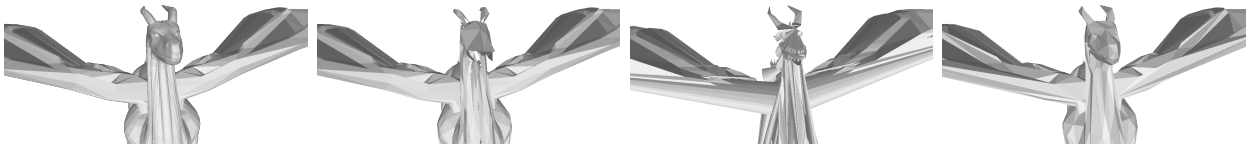
Textured Models	**FMPD**				
	$\boldsymbol{QEM_{ORIG}}$	$\boldsymbol{QEM_{BP}}$	$\boldsymbol{QEM_{TP}}$	$\boldsymbol{QEM_{BTP}}$	$\boldsymbol{QEM_{4VR}}$
Head	**0.312**	**0.312**	0.516	0.516	0.523
Jet	1.470	2.946	0.122	**0.096**	1.999
Airplane	**0.338**	0.990	$FAIL^*$	$FAIL^*$	1.423



(a) Original model (b) QEM_{ORIG} (FMPD = 1.039) (c) QEM_{BP} (FMPD = 1.921) (d) QEM_{4VR} (FMPD = 1.744)

Figure 2.13: Simplification of Car (manifold with boundaries) to approximately 10% its original number of faces, along with corresponding FMPD measures.

For the "RockerArm" (manifold model), the reduced mesh obtained with QEM_{4VR} and other QEM variants are perceptually similar (see Figure 2.5) and consequently corresponding FMPD measures are also similar (see Table 2.6). However, in the case of the "Car" model (manifold with boundaries), even though the reduced mesh obtained using QEM_{ORIG} has a lot of gaps near the windshield and side mirrors (See Figure 2.13), the corresponding FMPD metric is 1.039 (lower than QEM_{4VR}). Similarly, in case of the Dragon model (non-manifold), the distorted and incomplete reduced mesh obtained with QEM_{ORIG} has a lower FMPD metric.



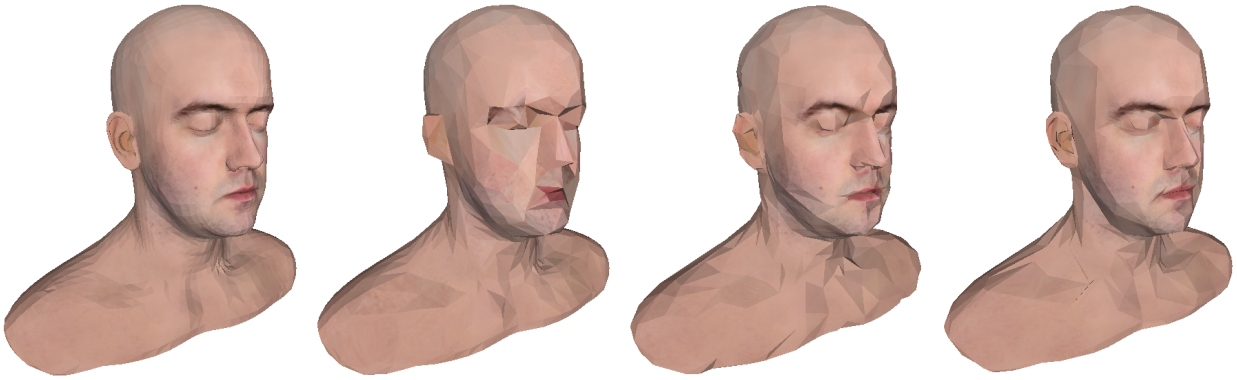
(a) Original model

(b) QEM_{ORIG} (FMPD = 0.185)

(c) QEM_{BP} (FMPD = 0.909)

(d) QEM_{4VR} (FMPD = 0.220)

Figure 2.14: Simplification of Dragon (non-manifold) to approximately 10% its original number of faces, along with corresponding FMPD measures.



(a) Original model

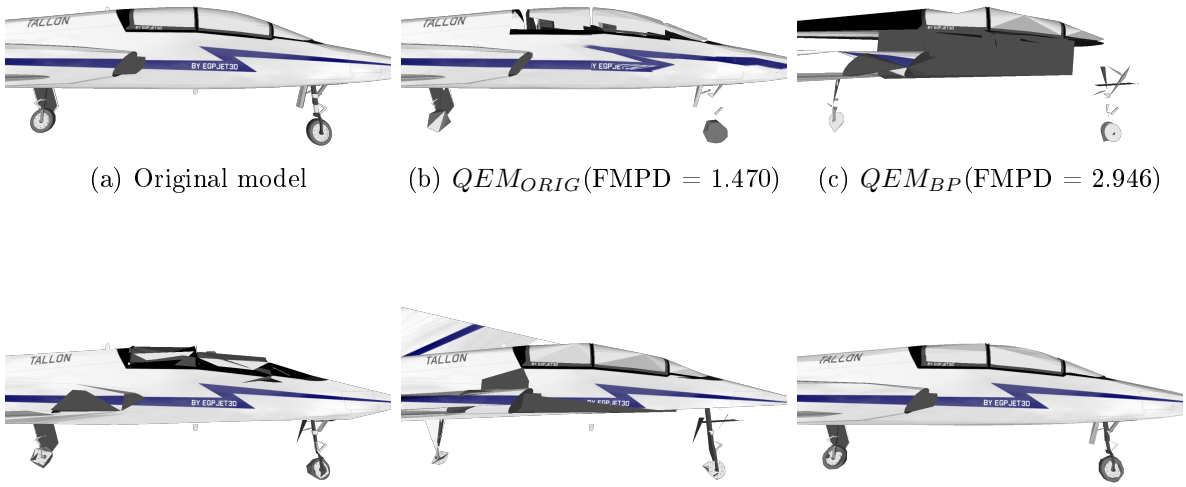
(b) QEM_{ORIG} (FMPD = 0.312)

(c) QEM_{TP} (FMPD = 0.516)

(d) QEM_{4VR} (FMPD = 0.523)

Figure 2.15: Simplification of Head (textured manifold with no boundaries) to approximately 10% its original number of faces, along with corresponding FMPD measures. As QEM_{BP} and QEM_{BTP} results are similar to QEM_{ORIG} and QEM_{TP} respectively, they are excluded for succinctness.

Furthermore, when the FMPD metric is applied to textured models, similar results are observed. For the textured "Head" model, the FMPD measure for QEM_{4VR} is higher than for other QEM variants even when the reduced mesh obtained using QEM_{4VR} has a better perceptual quality (see Figure 2.15, various sharp features such as the nose and ears are well maintained using QEM_{4VR}). Moreover, the FMPD measures for the low-poly models obtained using QEM_{ORIG} and QEM_{BTP} are lower than QEM_{4VR} despite the presence of a large number of gaps at boundaries and the corresponding visual quality not being satisfactory (see Figure 2.16).



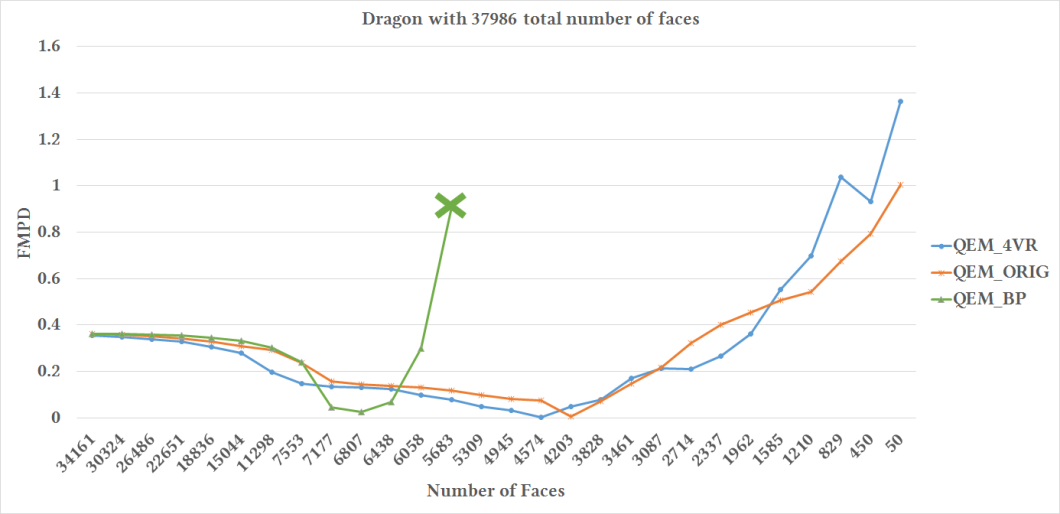
(a) Original model (b) QEM_{ORIG}(FMPD = 1.470) (c) QEM_{BP}(FMPD = 2.946)

(d) QEM_{TP}(FMPD = 0.122) (e) QEM_{BTP}(FMPD = 0.0963) (f) QEM_{4VR}(FMPD = 1.999)

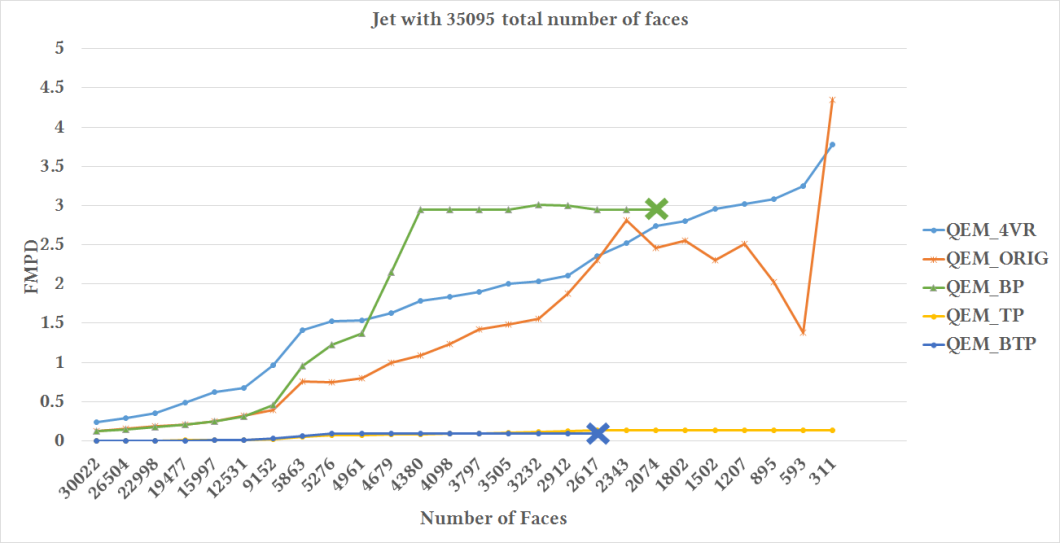
Figure 2.16: Simplification of Jet down to approximately 10% its original number of faces, along with corresponding FMPD measures.

Progressive Simplification: To further evaluate the effectiveness of the FMPD, we obtained various low-poly models with different levels of sparseness (i.e., with a different number of faces) for the Dragon and Jet models. For these low-poly models, we computed the FMPD. Figure 2.17 illustrates the behavior of the FMPD while reducing the number of faces for the Dragon and Jet models. Logically, as the number of faces reduces the perceptual distance from the original model should monotonically increase. However, FMPD has a contrasting behavior in different cases. As shown in Figure 2.17, the FMPD curve for various mesh simplification methods has anomalous characteristics with both non-textured and textured models.

To summarize, because the FMPD metric measures the difference in the global roughness as a perceptual distance, it does not take boundary gaps into account. Due to the inability of FMPD to incorporate boundary gaps and the completeness of the model while determining perceptual quality, it does not accurately represent the subjective visual quality. Therefore,



(a) For Dragon model



(b) For Jet model

Figure 2.17: FMPD with respect to the number of faces in simplified models. Termination of error curve indicates that no further poly count reduction can be achieved with the respective method.

for the low-poly mesh generated using QEM_{ORIG}, despite the presence of large boundary gaps, the FMPD metric is lower than compared to QEM_{4VR}. Although existing perceptual quality metrics can accurately measure degradation due to noise addition (e.g., watermarking), these methods are unable to accurately measure the subjective perceptual quality of simplified meshes.

2.6 Future JND Studies Evaluating Subjective Perceptual Quality

In section 4, we demonstrated that the objectively computed FMPD metric does not accurately reflect the subjective perceptual quality of our simplified meshes. This is due to the fact that the FMPD metric is based on the deviation in global surface roughness while subjective perceptual quality is dependent upon the Human Visual System (HVS) (Yang et al., 2005). Most objective perceptual quality metrics assume that visual quality increases as the number of vertices increase, but user studies have shown that not every set of vertices has a significant impact on visual quality (Cheng et al., 2006). More recently, Wang et al. (Wang et al., 2017) clearly demonstrated that traditional quality measures, which are linear in nature, do not account for the nonlinear aspects of human perception. Hence, in order to truly evaluate the perceptual quality of the QEM_{4VR} algorithm, we must rely on the HVS and conduct user studies on subjective perceptual quality.

As discussed in section 2.3, a JND methodology can be used to evaluate the visual qualities of multimedia (e.g., (Yang et al., 2005; Zhao et al., 2011; Hachicha et al., 2013; Wu et al., 2011)). However, there are three JND methodologies commonly used in the field of psychophysics: 1) the method of constant stimuli, 2) the method of limits, and 3) the method of adjustment (Qin et al., 2009). With the method of constant stimuli, a predetermined series of repeating stimuli are shown to the user, and the user is asked to respond if a difference is noticed (Watson and Fitzhugh, 1990). While highly accurate due to keeping subjects from anticipating changes in the stimulus, the method is considered inefficient (Qin et al., 2009;

Watson and Fitzhugh, 1990). With the method of limits, a stimulus is continuously increased or decreased until it is perceivable by the user (Qin et al., 2009). A common implementation of the method of limits is the staircase or up-down method, in which the stimulus is increased when it is not perceived and decreased when it is perceived (Watson and Fitzhugh, 1990). Finally, with the method of adjustment, the user can freely adjust the difference until it is not perceived (Qin et al., 2009).

Recent research by Qin et al. (Qin et al., 2009) has demonstrated that a one-up-two-down staircase method yields smaller JNDs than an adapted method of adjustment. Hence, we plan to employ the same method for evaluating the subjective perceptual quality of our QEM_{4VR} algorithm. In our future studies, participants will not be asked whether they see a difference between two meshes, but instead will be asked to indicate which mesh is visually higher quality. For each side-by-side comparison, both meshes will be simultaneously rotated one full cycle in a view-independent manner to allow all silhouettes to be examined, similar to the JND study conducted by Cheng and Boulanger (Cheng and Boulanger, 2005) on the effects of viewing distance on level of detail.

For each comparison, if the participant selects the incorrect mesh (i.e., the lower-poly mesh), the simplification difference (i.e., difference in polygon counts) will be increased by 5% of the original model's polygon count. This is the "one-up" aspect of the staircase method. When the participant selects the correct mesh (i.e., the higher-poly mesh), both meshes will be redisplayed in a random order. If the participant again correctly selects the higher-poly mesh, the simplification difference will be decreased by 5% for the next pair of meshes. This is the "two-down" aspect.

We have planned for conducting two JND studies to evaluate the subjective perceptual quality of our QEM_{4VR} algorithm. The first JND study will focus on non-textured models and will have participants complete the one-up-two-down staircase method for QEM_{ORIG}, QEM_{BP}, and QEM_{4VR}. The second JND study will focus on textured models and will have

participants complete the method for QEM_{TP}, QEM_{BTP}, and QEM_{4VR}. For both studies, we have prepared simplified meshes for six models (two manifolds with no boundaries, two manifolds with boundaries, and two non-manifolds). We hypothesize that the results of these studies will demonstrate that QEM_{4VR} yields a larger JND than the other algorithms for both non-textured and textured models, which would indicate that QEM_{4VR} affords better subjective perceptual quality. We are currently awaiting Institutional Review Board (IRB) approval to begin conducting our JND studies.

CHAPTER 3

REAL-TIME, CURVATURE-SENSITIVE SURFACE SIMPLIFICATION USING DEPTH IMAGES[1]

3.1 Introduction

In recent times, handheld mobile Virtual Reality (VR) devices such as Samsung Gear VR, Occulus Rift, Microsoft HoloLens are becoming increasingly popular due to the great immersive VR experience they provide. These devices coupled with game engines such as Unity3D and Unreal Engine provide a feasible and affordable VR experience for applications in diverse areas. In combination with these devices, the availability of low-cost, off-the-shelf RGB-D cameras, such as Microsoft(MS) Kinect, has enabled numerous real-time applications in the domain of virtual training for various tasks, collabrative visualization and 3D Tele-immersion (3DTI) (Kreylos, 2005; Arworks, 2015; Microsoft, 2016), etc. These real-time applications need to balance visual quality, rendering latency, power consumption and battery life to achieve an optimal immersive experience (Microsoft, 2015). Mobile VR devices are also constrained with low processing power and onboard memory leading to an upper bound on the data size. For example, Samsung Gear VR has a working limit of 50K-100K polygons (Pruett, 2015) and Microsoft HoloLens has essential performance limit of < 900 MB for memory. Moreover, for Samsung Gear VR, it is recommended to keep the 50k polygon count per frame for the interactive and immersive experience. Each frame can have several models leading to reduced vertex count per model. Further, game engines such as Unity3D can handle at max 65536 vertices per model due to lack of 32-bit indexing support (Unity3D, 2012). Though, Unity3D handles the vertex limit by splitting the mesh into subparts, the frame rate reduces significantly with increasing polycount resulting in the

[1] ©2018 IEEE. Reprinted, with permission, from K. Bahirat, S. Raghuraman and B. Prabhakaran, "Real-Time, Curvature-Sensitive Surface Simplification Using Depth Images," in IEEE Transactions on Multimedia, vol. 20, no. 6, pp. 1489-1498, June 2018. DOI: 10.1109/TMM.2017.2769447

(a) (b)

Figure 3.1: For a person scanned using Kinect, a) Original dense textured mesh with 52885 vertices, and b) sparsely approximated textured mesh with 5768 vertices obtained using our method.

degraded immersive experience. On the other hand, typical RGB-D cameras such as Microsoft (MS) Kinect generates a dense depth data consisting of maximum 300K vertices @30 fps. Even if the input is a compressed depth stream obtained using transmission standards such as 3D-HEVC (Tech et al., 2012), the uncompressed stream still gives a dense sampling. Hence, matching the limitation of handheld mobile VR devices and rendering engines while maintaining a realistic, immersive experience is a challenging task.

One possible solution is to reduce the overall data size by obtaining a sparser approximation of the 3D scenes or objects. While getting such a sparser approximation, it is also important to keep in mind that the finer details of the scene/object should be retained to maintain high fidelity of the system, as shown in Figure 3.1. Surface simplification is the notion of reducing the data size which is achieved by removing some vertices/points representing the object/scene. Traditional mesh and point cloud simplification methods used for surface simplification need high execution time and are inadequate for real-time applications.

Problem Statement: Given a raw depth map R which describes the scanned surface S using a dense set of points X, find $X_{Reduced} \subset X$ such that surface $S_{Reduced}$ reconstructed

from $X_{Reduced}$ by triangulation approximates S i.e., $|S - S_{Reduced}| < \varepsilon$ while maintaining $|X_{Reduced}| < n$ where n is required number of samples in a sparse representation of surface and ε is very small. Characteristics of the points in $X_{Reduced}$ should be such that $S_{Reduced}$ preserves the overall object shape maintaining its integrity while preserving finer details.

3.1.1 Proposed Approach

In this paper, we introduce a depth image based approach to sparsely sample a surface for real-time mesh generation and visualization. A Curvature Sensitive Surface Simplification - CS^3 operator is proposed, which: (i) utilizes the grid structure of depth image to obtain neighborhood information as a pre-processing step; (ii) assigns the importance to each point based on surface variation; (iii) and uses these importance values for selecting a detail-preserving sparse sampling of a surface. For each of the later two steps, we explore various possible vision techniques. Based on a comparative study of these techniques, we design the CS^3 operator that will create a area-weighted normal map of the depth image. Using this map, CS^3 then computes the curvature-based importance of each point in the depth image and performs a restrictive sampling based on the order of importance of the points. We also modify the 2D sweep-line constrained Delaunay triangulation to generate 3D meshes from the sparsely sampled surfaces given by the CS^3 operator. Formulation of the 3D surface simplification problem in 2D domain makes the proposed method computationally very efficient. The performed experimental study demonstrates that the proposed method can achieve a high reduction in the data size without significant degradation in visual quality, in real-time.

Principal Contributions of this paper are:

- Mapping the 3D surface simplification problem to 2D domain by utilizing the depth image.

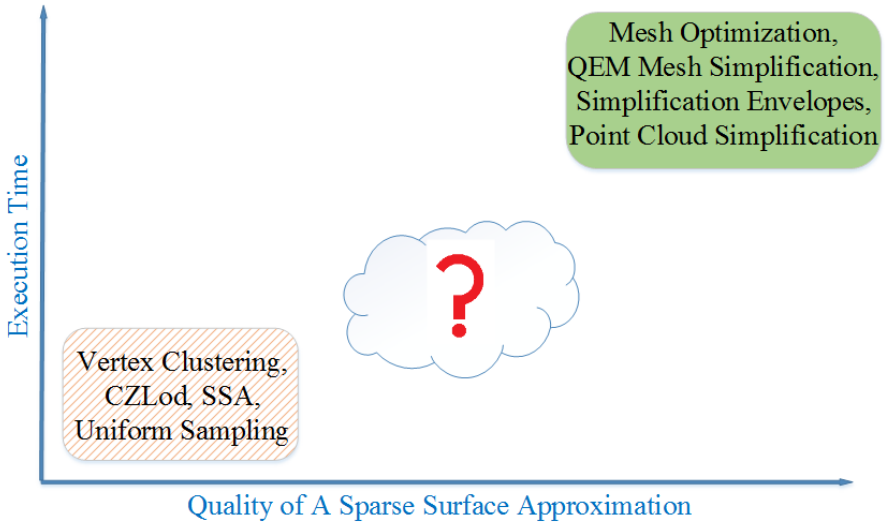


Figure 3.2: Classification of various surface simplification methods.

- A real-time depth image based sparse sampling method for 3D surfaces, which is sensitive to the underlying surface curvature.

- A modified 3D constrained Delaunay triangulation that allows discontinuities and self-occlusion in the entire real world scene data.

- Elimination of sensor noise at object boundaries as a byproduct of the sampling method and modified triangulation.

- User-controlled sparseness is allowed which can potentially mitigate the limitations of handheld VR devices and rendering engines.

- Implicit preservation of surface properties such as texture coordinates by keeping the point position unaltered.

- Distance-dependent simplification of the complete scene

3.2 Related Work

Various simplification algorithms have been previously proposed to achieve a sparser surface approximation (please refer to Cignoni et al. (Cignoni et al., 1998) and Luebke (Luebke, 2001) for two excellent surveys). Most of the methods are designed to work offline to generate high quality low-poly meshes. Very few methods are developed for real-time performance. Figure 3.2 illustrates the various categories for simplification algorithms. A key requirement for real-time VR application is that the simplification algorithm must perform in real-time while maintaining the acceptable visual quality. Hence, in this section, we distinguish between algorithms designed to work offline and online algorithms.

3.2.1 Non Real-time Methods

Mesh Simplification:

One of the earliest mesh simplification algorithms was the decimation algorithm presented by Schroeder et al. (Schroeder et al., 1992). For each vertex of the model, the decimation algorithm determines whether a vertex can be removed without violating the topology of the local neighborhood of faces. If the resulting surface would be within a user-defined distance of the original mesh, the algorithm removes the vertex and all of its neighboring faces. The resulting hole is then re-triangulated. The decimation algorithm makes multiple passes over the vertices of the model until it cannot remove any more vertices without violating the pre-defined distance criterion. The user-defined distance has a major impact on both the lower bound and fidelity of the algorithm as well as the execution time (Garland and Heckbert, 1997). Hoppe et al. (Hoppe et al., 1993) presented a mesh optimization algorithm that generates a topologically similar sparse mesh of the original mesh. To produce sparse meshes, Hoppe et al. (Hoppe et al., 1993) suggested three types of mesh transformations to apply to the original mesh: edge collapse, edge split, and edge swap. The mesh optimization algorithm randomly uses these transformations to produce new candidate manifold meshes,

and then picks the candidate that minimizes an energy function that represents the error of the mesh. Hoppe (Hoppe, 1996) later defined the concept of a progressive mesh, which represents a manifold mesh as a sequence of edge collapses. The original mesh can be retained by applying a series of vertex splits to the simplified mesh. Hoppe (Hoppe, 1996) also defined a new energy function to better represent mesh complexity and maintain a higher degree of fidelity.

Cohen et al. (Cohen et al., 1996) presented another approach to simplifying manifold meshes by using two offset copies of every mesh surface, called simplification envelopes. These offsets were used to ensure that a simplified surface would remain within the volume formed by the envelopes. Cohen et al. (Cohen et al., 1996) presented two approaches for iteratively removing triangles or vertices from the original mesh and re-triangulating any resulting holes.

Perhaps, the most common approach to simplifying non-manifold meshes is to use a quadric error metric (QEM) (also referred as QSlim), which was first presented by Garland and Heckbert (Garland and Heckbert, 1997). The QEM of a vertex is a 4 x 4 matrix that represents the sum of the squared distances from the vertex to the planes of adjacent faces. When a vertex is merged with another vertex within a user-defined distance threshold, the error introduced can be computed as the sum of the QEMs of the vertices being merged, which becomes the QEM of the new vertex. When merging vertices, the QEM algorithm keeps a sorted priority queue of all candidate vertex pairs based on their merged QEM. The algorithm removes the vertex pair with the lowest error from the top of the queue, merges the vertices, and then updates the errors of all vertex pairs involving the merged vertex. By using the QEM, this approach yields simplified meshes that are relatively high fidelity, even at drastic levels of simplification (Luebke, 2001). Garland and Heckbert (Garland and Heckbert, 1997) also extended their approach to preserve boundaries by defining a boundary constraint plane passing through each boundary edge. In their later work, Garland

and Heckbert (Garland and Heckbert, 1998) generalized the original QEM to also handle surface properties. To handle normals or colors, the original 4 x 4 error matrix can be extended to a 6 x 6 matrix that contains the error of the vertexâĂŹs normal (abc) or its color (rgb). Hoppe (Hoppe, 1999) also presented a generalized QEM that required less storage space than that of Garland and Heckbert (Garland and Heckbert, 1998) by using a wedge-based mesh data structure. Along with his generalized QEM, Hoppe (Hoppe, 1999) also proposed a memoryless simplification algorithm and a volume-preserving algorithm. Lindstorm and Turk (Lindstrom and Turk, 1998) proposed a fast, memory efficient polygonal simplification (FMEPS) that solely utillizes the current approximation to make decision. The selection of the edge for contraction and resultant vertex position are determined by linear constraint preserving the volume of the tetrahedron formed between the estimated vertex and neighboring faces. More recently, Ovreiu (Ovreiu, 2012) presented two quadric error metrics capable of handling surface attributes.

Point Cloud Reduction: Boissonnt et al. (Boissonnat and Cazals, 2001) proposed one of the initial point cloud simplification algorithm that selects a random initial subset of the point cloud and uses it's 3D Delaunay triangulation to define a signed distance function over the selected set. If this implicit function does not satisfy the user-defined approximation tolerance, then additional points are selected to enlarge the initial set. Linsen et al.(Linsen, 2001) measured an information content associated with every point in terms of a local curvature and RGB color changes and subsequently removes points featuring the lowest entropy. Pauly et al. (Pauly et al., 2002) suggested the adaptation of various widely used mesh simplification methods for the point cloud simplification scenario. Alexa et al. (Alexa et al., 2003) proposed a method that uniformly reduce point cloud by removing points contributing least to the moving least square (MLS) representation of the surface. Moenning et al. (Moenning and Dodgson, 2003) suggested a farthest sampling that places the next sampled point in the middle of the least known area.

3.2.2 Real-time Methods

The vertex clustering algorithm presented by Rossignac and Borrel (Rossignac and Borrel, 1993) is one of the computationally efficient method providing a real-time performance. Vertex clustering algorithm creates a three-dimensional grid enveloping the original mesh. For each cell in the grid, it computes a representative vertex and collapse all the vertices in the cell to the representative vertex. The resolution of the grid determines the fidelity of the resulting simplified mesh. However, because the approach does not guarantee the amount of error introduced by the simplification, vertex clustering algorithms are often visually less pleasing than other mesh simplification algorithms (Luebke, 2001).

Various approaches are designed for the 3D data transmission over the network (Wu et al., 2011; Cheng and Boulanger, 2006). Wu et al. (Wu et al., 2011) designed a 3D tele-immersive system that uses a bisection triangulation algorithm for creating the different levels of sparseness termed as "CZLod". Starting with the first four triangles between center and corners, it splits each triangle into two smaller triangles until the depth variation in a triangle is less than a predefined threshold. Due to the depth variation criterion, if the planar object is at an angle to the viewing direction then it creates the nonoptimal dense sampling. Cheng et al. (Cheng and Boulanger, 2006) presented a single pass surface simplification algorithm using a scale-space analysis (SSA) and zero-crossing detection. The algorithm (referred as SSA) first transform the object's surface to a 2D cylindrical coordinate system by determining the medial axis of the object. Next, it applies Laplacian of Gaussian (LoG) operator followed by a zero-crossing detection to select points. Different levels of the sparseness are obtained by varying the standard deviation of Gaussian Filter. The scale-space analysis based method considers only the local neighborhood of point during selection and does not distribute the point density over the entire surface resulting in the loss of the overall surface shape. Further, the requirement of medial axis determination makes it unsuitable for simplifying the complete scene. Hou et al. (Hou et al., 2015) suggested a surface reconstruction algorithm that

randomly selects a subset of original set of points using the Poisson disk sampling. Based on existing selection, the algorithm selects a next point atleast some predefined distance (called as a Poisson disk's radius) away from existing points. Next, the algorithm generates an approximate Voronoi diagram for the sampled points to obtain a surface reconstruction. As the selection method does not consider local surface characteristics, this method may not preserve finer details.

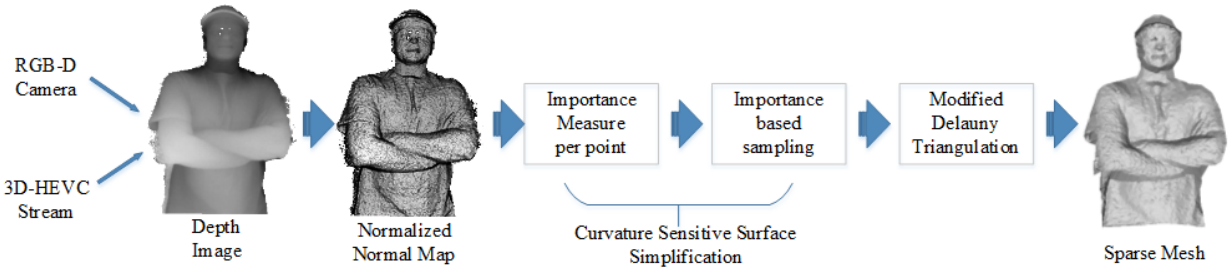


Figure 3.3: Block diagram illustrating the pipeline of the proposed curvature-sensitive surface simplification.

3D Video Compression: 3D video transmission standards such as 3D-HEVC effectively reduce transmission latency (Tech et al., 2012). Various depth map coding methods (Mehrotra et al., 2011; Kim and Ho, 2007; Oh et al., 2011; Gautier et al., 2012) are proposed utilizing the fact that depth maps are more homogeneous compare to texture and depth of a given pixel can be predicted from neighboring pixels. However, decompressing the compressed streams would still yield a densely sampled data.

3.2.3 Drawbacks of Existing Methods

- Most of the mesh simplification methods are iterative and computationally expensive.
- To apply mesh simplification methods, first a dense triangulation need to be computed from the depth image that further increases the complexity.

- Point cloud reduction approaches can not avail connectivity information. Hence, most of these methods need to apply the k-nearest neighborhood for knowing local surface properties making them computationally inefficient.
- Vertex clustering methods are computationally efficient, but they cannot adapt to non-uniformities in the sampling distribution.
- Methods such as CZLod and scale-space analysis based surface simplification provide a real-time performance but they do not preserve the finer details of a surface resulting in the degradation in a visual quality.
- A 3D compressed stream, after decompression, still provides a dense data at rendering side.

Above-mentioned drawbacks motivate the design of a single-pass method that will achieve a trade-off between execution time and visual quality.

3.3 Curvature Sensitive Surface Simplification

To achieve the trade-off between visual quality and execution time, we propose a single-pass simplification approach that works as follows:

- As a pre-processing step, we first get a grid-based surface representation using area-weighted normal maps.
- Next, we apply a curvature-sensitive surface simplification operator (CS^3) to select the reduced set of points representing the surface. Selected points are triangulated to obtain a sparser surface approximation.

Figure 3.3 illustrates the pipeline of the proposed method. The input can be a raw depth image and registered texture from RGB-D cameras or an uncompressed data from any 3D compressed stream such as the 3D-HEVC streams.

3.3.1 Grid-based Surface Representation

As a pre-processing step, we obtain a 2D representation of the given surface where the connectivity information can be obtained implicitly through the image grid structure. To achieve this goal, we first compute the normal map from the depth image and intrinsic camera parameters. Next, we compute the area-based normalization of the normal map.

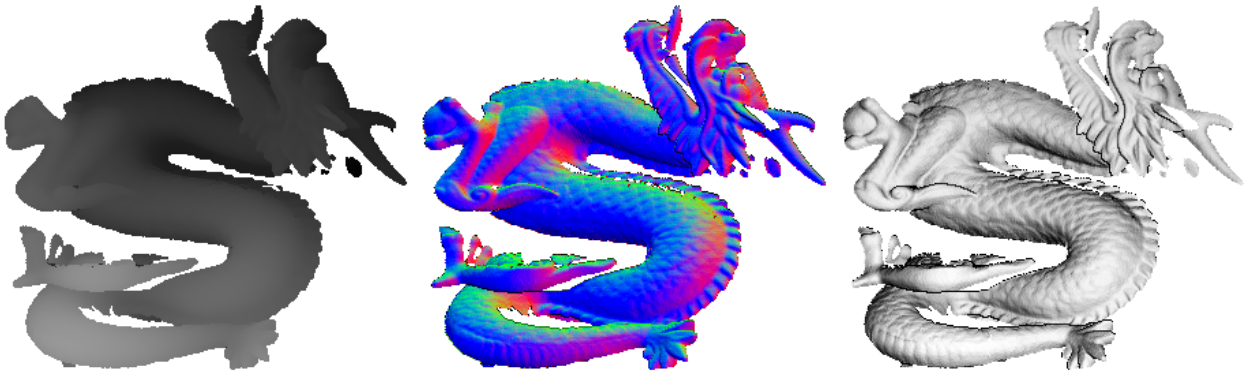
Normal Map Computation: The proposed simplification algorithm takes a raw depth map R and camera calibration matrix K obtained by Kinect-like RGB-D cameras as an input. A raw depth map R provides the depth measurement $R(\mathbf{u}) \in \Re$ at each image pixel $\mathbf{u} = (u, v)^T$ in the image domain $\mathbf{u} \in \Re^2$ such that $\mathbf{p} = R(\mathbf{u})K^{-1}\dot{\mathbf{u}}$ is a metric point measurement in the sensorâĂŹs frame of reference (Izadi et al., 2011). Similarly, all the points are projected back in the sensor's frame of reference to obtain a vertex map V. The vertex map provides the surface measurement on a regular grid. The regular grid structure of the vertex map provides the local connectivity information without actually performing the triangulation of the entire set of points. Hence, using the cross product between neighboring vertices; we compute corresponding normal vectors,

$$N(\mathbf{u}) = (V(u+1, v) - V(u, v)) \times (V(u, v+1) - V(u, v)) \tag{3.1}$$

In order to reduce the problem to 2D for efficient computation, we compute L2-norm of each value in the normal map and normalize it by dividing with the product of squared lengths of two edges formed between the current point and its neighboring points on the map, as follows:

$$\begin{aligned} V_{12}(\mathbf{u}) &= (V(u+1, v) - V(u, v)) \\ V_{13}(\mathbf{u}) &= (V(u, v+1) - V(u, v)) \\ A(\mathbf{u}) &= \frac{|N(\mathbf{u})|}{|V_{12}(\mathbf{u})|^2 * |V_{13}(\mathbf{u})|^2} \end{aligned} \tag{3.2}$$

Here, the denominator in (2) incorporates the weighing of values based on the area of a triangle formed by these three points. Figure 3.4(a), 3.4(b) and 3.4(c) show the depth map,



(a) (b) (c)

Figure 3.4: For Stanford dragon model, a) Depth Image $R(\mathbf{u})$, b) Colored rendering of normal map $N(\mathbf{u})$, and c) Area-weighted normal Map $A(\mathbf{u})$.

normal map N and area-weighted normal map A for the dragon model from Stanford range dataset. It can be seen that the surface variation is very well captured in A as compared to the original depth map R itself. Definition of A provides us a surface representation in terms of function $A(u, v)$: $\Re^2 \rightarrow \Re$.

3.3.2 Curvature-Sensitive Surface Simplification (CS^3) Operator

Given a grid-based 2D surface representation, next we sparsely sample the object surface. We propose a CS^3 operator for a single-pass surface simplification approach that operates in two steps: (i) compute the curvature sensitivity by determining the importance of each point in representing the underlying surface; (ii) select points as a candidate vertices based on their importance.

Curvature-Sensitivity Computation

It can be observed that the points on a boundary, on ridges and at corners aptly represent the curvature variations of the underlying surface. In order to preserve finer details, these curvature-sensitive points should be included in the reduced set of points. For identifying such key feature points of the surface, let us consider that the given surface is approximated

by a polygonal mesh. If the normals of faces incident on a vertex are in the same direction, then we can consider that a given vertex is lying on a planar surface patch. Hence, it can be approximated by its projection on a plane passing through its neighboring vertices without introducing any significant error. On the other hand, if the normal direction is changing significantly for faces that are incident on a vertex then that vertex is important to approximate the surface with the less error. Hence, variation in normal directions in a local surface patch characterizes the surface curvature in that region. This local surface curvature is well captured by Hessian operator (Besl and Jain, 1986) on a normalized map, which computes a second order partial derivative of the area-weighted normal map A(u,v) such as $\frac{\partial^2 A}{\partial u^2}$, $\frac{\partial^2 A}{\partial v^2}$ and $\frac{\partial^2 A}{\partial u \partial v}$ at each point $\mathbf{u}$. Using these second order partial derivatives, a Hessian matrix M is computed at each point $\mathbf{u}$ as follows:

$$M(\mathbf{u}) = \begin{bmatrix} \frac{\partial^2 A}{\partial u^2} & \frac{\partial^2 A}{\partial u \partial v} \\ \frac{\partial^2 A}{\partial v \partial u} & \frac{\partial^2 A}{\partial v^2} \end{bmatrix} \tag{3.3}$$

Importance Score Computation

We define the importance of a point as a score, $\Omega(\mathbf{u})$, which will represent the contribution of that point to the overall shape or finer details of the object. The importance of a point should be based on the associated local surface curvature. $\Omega(\mathbf{u})$ can be computed in few different ways. In this paper, we explore two possibilities for computing $\Omega(\mathbf{u})$.

(a) Laplacian-based Importance: Given a Hessian matrix M, one possible option for the importance measure can be the average variation in the local surface patch. The average variation in the local surface patch can be easily computed using a Laplacian operator which is described as follows:

$$\Delta A = \nabla^2 A = \frac{\partial^2 A}{\partial u^2} + \frac{\partial^2 A}{\partial v^2} \tag{3.4}$$

Corresponding importance measure can be given as:

$$\Omega(\mathbf{u}) = \Delta A(\mathbf{u}) \tag{3.5}$$

(b) Minimum Curvature-based Importance: Another possibility is to utilize the principal curvature as an importance measure. Eigenvalues of Hessian matrix play a very crucial role in determining characteristics of the underlying surface and consequently the surface curvature. Let $\lambda_1(\mathbf{u})$ and $\lambda_2(\mathbf{u})$ be the Eigenvalues of $M(\mathbf{u})$. If both Eigenvalues are too small, then the region is planar. If one Eigenvalue is higher than the other one, then the curvature in the direction of the Eigenvector corresponding to the higher Eigenvalue is maximum. In this case, a point can be assumed to be on the ridge or the edge. If both the Eigenvalues are high and positive, it represents the local maximum, and the corresponding point can be considered as a corner point. We can observe that the points with a higher minimum Eigenvalue can be considered to be lying in the high curvature region. It should be noted that if a point lies on the part of the surface which is locally concave, then the minimum Eigenvalue will be negative. Hence, to nullify the effect of direction of surface variation, we consider the absolute of the minimum Eigenvalue. Therefore, we define the importance measure as:

$$\Omega(\mathbf{u}) = |min(\lambda 1(\mathbf{u}), \lambda 2(\mathbf{u}))| \tag{3.6}$$

Importance Score based Sampling:

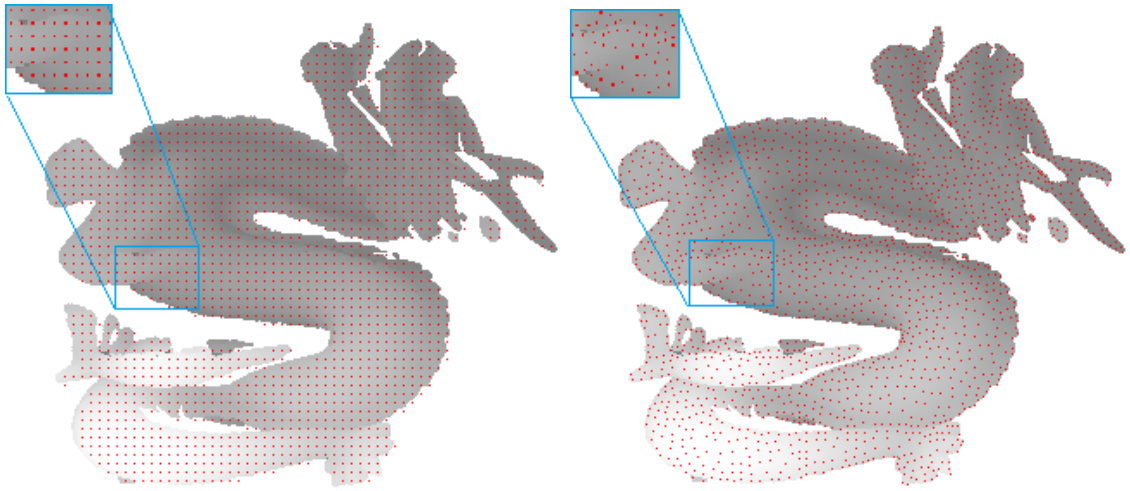
We propose a sampling mechanism adaptive to local curvature of the underlying surface. This sampling mechanism will traverse through the entire list of points just once and select the set of points that are important to represent the surface at the required level of sparseness. Surfaces with high variation requires more points and low variation can be represented by lesser points. Hence, high curvature regions require more dense sampling for proper

representation; whereas, sparse sampling is adequate for the planar region. We propose two deterministic time coherent sampling methods:

(a) Sampling by Non-Maximum Suppression: It is well known that, in Canny Edge detector (Canny, 1986), non-maximum suppression is achieved by comparing the edge gradient magnitude at the current pixel with two immediate neighbors in the positive and negative edge directions. Deriving inspiration from the Canny Edge detector, we propose to utilize the principal curvature directions for non-maximum suppression. Note that, principal curvature directions can be easily obtained in terms of Eigenvectors of a Hessian Matrix. Suppose, v_{min} and v_{max} are Eigenvectors of a Hessian matrix M corresponding to Eigenvalues λ_{min} and λ_{max} respectively. Then, for each pixel p, we compare those Eigenvalues with those of its neighboring pixels in the positive and negative principal curvature directions. We refer to the set of these neighboring pixels as a curvature neighborhood. If the importance score $\Omega(\mathbf{u})$ at the current pixel is maximum compared to all the points in the curvature neighborhood, then it is preserved else its value is suppressed to zero. The scope of the curvature neighborhood is selected as w_{max} number of points in v_{max} direction and w_{min} number of points in v_{min} direction. To determine the values of w_{max} and w_{min}, we utilize the fact proven in the approximation theory (Simpson, 1994)(Heckbert and Garland, 1999) that the L_2 optimal approximation to the smooth surface have asymptotic stretching ratio proportional to $\sqrt{\frac{|\lambda_{max}|}{|\lambda_{min}|}}$. Hence, given a 2D sampling interval $s = \sqrt{(w * h)/n}$ where w and h are the width and height of bounding box enclosing the object and n be the user-specified size of reduced set of points, we compute w_{max} and w_{min} as follows:

$$w_{max} = s, w_{min} = \sqrt{\frac{|\lambda_{max}|}{|\lambda_{min}|}} * w_{max} \tag{3.7}$$

Note that, the values of w_{max} and w_{min} change according to principal curvature value at that point resulting in a curvature adaptive sampling.



(a) (b)

Figure 3.5: For Stanford dragon model, 4% sampled data a) using uniform sampling and b) proposed surface sampling.

(b) Importance Order-based Sampling: Another possible approach is to preserve the finer details of the surface by deciding the order of point selection based on the importance values. It will result in the selection of high importance points first and consequently favors the high curvature features. While favoring the high curvature points, the care should be taken to maintain the integrity of the object such that the sampling mechanism does not pick up only high curvature points. Otherwise, it may result in the loss of overall object shape at a very high level of sparseness. To achieve this objective,

- We restrict how sparsely one can sample based on the user-requested size of the reduced point set.
- We change the density of sampling adaptive to the surface by introducing the notion of *repulsive force* exerted by the selected point.

All the points are sorted in decreasing order of importance measure. We pick the point with the highest importance first. The selected point exerts a repulsive force on its surrounding points that will block further sampling in the vicinity of the selected point. The

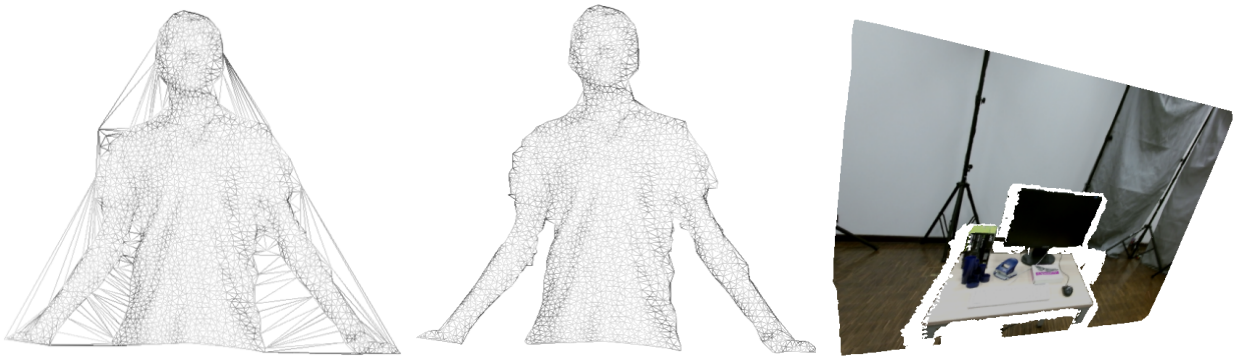
size of the region affected by the repulsive force is determined by the importance value of the selected point, required level of reduction and the overall extent of the object in the depth map. Let w and h be the width and height of bounding box enclosing the object in depth map and n be the user-specified size of reduced set of points. Then the maximum possible length of radius of squared window representing the region affected by repulsive force will be given by $Rd_{max} = \sqrt{(w * h)/n}$. For any selected point, the radius of the affected region is given as:

$$Rd(\mathbf{u}) = \left(1 - \frac{\Omega(\mathbf{u}) - \Omega_{min}}{\Omega_{max} - \Omega_{min}}\right) * Rd_{max} \tag{3.8}$$

This definition of a window size allows to adapt itself dynamically to the curvature of the underlying surface. If the selected point has a high importance, then the affected region due to the repulsive force exerted by the point will have a smaller area. On the other hand, if the selected point has low importance then the affected region of corresponding repulsive force will be larger. This facilitates the dense sampling in high curvature region and sparse sampling in a planar region. The dependency of Rd_{max} on n and application of repulsive force by selected points guarantee that points are selected from all over the surface without concentrating sampling density in high curvature region even though overall sampling strategy favors the high importance points more. Figure 3.5(a) and 3.5(b) show the reduced set of points obtained by the uniform sampling and by the proposed surface sampling approach on the backdrop of the depth image.

3.4 Triangulation

For the visualization, the set of reduced points needs to be triangulated to obtain a mesh and to render along with the texture information. Generating a mesh in real-time is in itself a challenging task. The problem becomes further complicated due to sparse data points with no neighborhood information. The majority of the techniques used for generating



(a) (b) (c)

Figure 3.6: For a person scanned using Kinect, a) triangulation proposed in (Domiter and Zalik, 2008), and b) our modified triangulation; c) A sparse mesh (10% of original size) for the desk model from CoRBS dataset obtained using our method.

a mesh in 3DTI, such as (Raghuraman et al., 2013a) work on a dense 2D image having neighborhood information in the form of the grid structure. Due to sparse sampling for surface simplification, the connectivity information is lost, making these fast 2D meshing techniques unsuitable. Other approaches such as Poisson surface reconstruction, marching cube algorithm assume closed nature of sampled points, making them unsuitable for open surfaces scanned by RGB-D cameras such as Kinect.

Modified Delaunay Triangulation: We base our triangulation on fast sweep-line constrained Delaunay triangulation (CDT) (Domiter and Zalik, 2008). CDT meshes sparse data irrespective of point distribution very quickly. As this method does not have any neighborhood information incorporated during triangulation, it may end up formulating unwanted triangles between different parts of the object resulting in a situation as shown in Figure 3.6(a). To resolve this issue, we modify the approach proposed in (Domiter and Zalik, 2008) by incorporating the information about original depth map to validate the generated triangles. To check the validity of a triangle, we compute the centroid of triangle and project it on the depth map followed by checking the availability of valid depth measurement at a projected point. If there is no valid depth measurement at the projected point corresponding

to the centroid, then the triangle is considered to be invalid. It adds a little overhead in execution time needed for meshing.

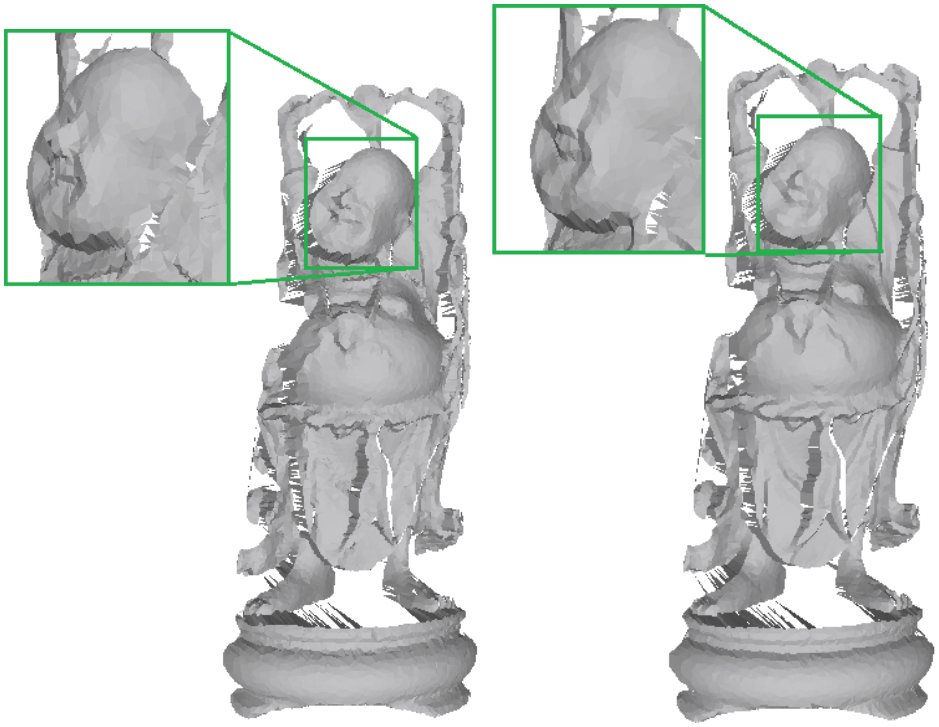
Triangulation of a Scene: When the proposed simplification is applied to the entire scene, the sweep-line constrained Delaunay triangulation has to be further modified. The scene might consist of different objects situated at different distances from the camera. One possible heuristic is to assume that the points belonging to different objects will be at a larger distance compared to points belonging to the same object. Using this fact, we modify the above triangulation algorithm to check the validity of a triangle based on the maximum possible distance between points. If the distance between any pair of points in a given triangle is greater than a certain threshold (determined empirically), then the triangle is declared as an invalid triangle. As this is a heuristic approach, objects which are in a close vicinity might get meshed together. For example, a book on a desk gets meshed together with the desk as shown in Figure 3.6(c).

3.5 Evaluation of CS^3

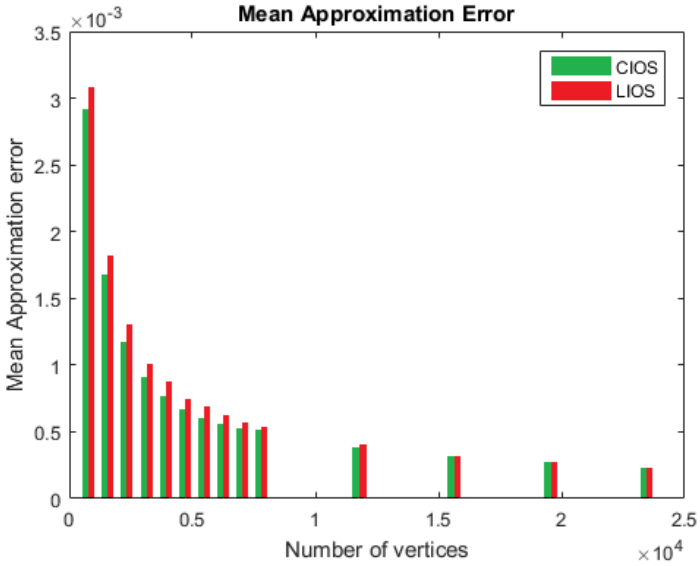
We performed detailed analysis of proposed methods with the goal of deciding a suitable method for real-time surface simplification. All the algorithms are implemented in C++ and experiments are run on a CPU with Intel(R) Core(TM)i7-5820K with 3.30GHz speed and 32GB internal RAM.

Datasets: We evaluated the performance of proposed methods across a variety of datasets scanned using multiple scanners to confirm the generality and the robustness of methods. The datasets used are as follows:

- ***D1:*** Stanford Range Dataset scanned using Cyberware 3030 MS (Levoy et al., 2005)
- ***D2:*** CoRBS Dataset scanned by MS Kinect (Wasenmüller et al., 2016)
- ***D3:*** Human models dataset scanned by MS Kinect



(a) (b)



(c)

Figure 3.7: For the Stanford Buddha, a sparser mesh (10% size) obtained using a) ***LIOS***, and b) ***CIOS***; c) METRO versus number of vertices for various sparser meshes of Stanford Buddha obtained using ***LIOS*** (yellow) and ***CIOS*** (blue).

The datasets include scanned scenes, scanned human models and other objects, both with and without texture. See Table 3.1 for a summary of the 3D models.

Table 3.1: Attributes of Datasets used for experiments.

Dataset	**Model**	**Faces**	**Vertices**	**Type**
Stanford Datsset	Dragon	82649	42641	Non-textured Model
	Bunny	79312	40256	Non-textured Model
	Buddha	151838	78056	Non-textured Model
CoRBS Dataset	Desk	3042969	155275	Textured Scene
	Human	296459	150753	Textured Scene
	Cabinet	331223	166943	Textured Scene
Human models	Person 1	88294	44931	Textured Model
	Person 2	69892	35822	Textured Model
	Person 3	66704	33898	Textured Model

We first compared the different possible options for importance score $\Omega(\mathbf{u})$ computation and the $\Omega(\mathbf{u})$-based sampling. Possible options are:

1. ***LNMS***: Laplacian based importance, sampling by non-maximum suppression.

2. ***LIOS***: Laplacian based importance, importance order-based sampling.

3. ***CNMS***: Minimum curvature-based importance, sampling by non-maximum suppression.

4. ***CIOS***: Minimum curvature-based importance, importance order-based sampling.

We analyzed the performance of above options in terms of how well they approximate the original surface using the one-sided approximation error. We utilized a METRO tool (Cignoni et al., 1996), which searches for each point on the original mesh, its closest point on the simplified triangular mesh and computes a Hausdorff distance between them as an error. The average of computed errors, called *mean approximation error* (henceforth referred as METRO) which is normalized with respect to the diagonal length of the original model's

Table 3.2: Statistics of execution time taken by different methods for Dragon, Bunny and Buddha model.

Model	# Input Points	# Output Points	***LNMS*** Time (ms)	***LIOS*** Time (ms)	***CNMS*** Time (ms)	***CIOS*** Time (ms)
Dragon	42641	4264	100	38	94	35
Bunny	40256	4026	91	40	88	35
Buddha	78056	7806	161	54	154	51

bounding box and is used as a final error measurement. We further analyzed their computation time and suitability for real-time applications. Table 3.2 enlists the execution time for all methods. It can be observed that the methods utilizing the sampling by non-maximum suppression require significantly higher execution time because of additional curvature direction computation based on Eigenvectors of a Hessian matrix. ***LIOS*** and ***CIOS*** meet the requirement of low execution time for real-time applications. Hence, we further compare them in terms of mean approximation error.

Figure 3.7(a) and 3.7(b) show sparser approximations for a Buddha model obtained using ***LIOS*** and ***CIOS***. It can be seen that the details such as the nose, mouth are well approximated with ***CIOS***. It is additionally verified by a quantitative measure METRO as shown in Figure 3.7(c). ***LIOS*** utilizes Laplacian as an importance measure which is the sum of two Eigenvalues of a Hessian matrix. Due to additive nature of this importance measure, the critical point lying on the locally concave part of the surface might be considered as less important. Hence, ***LIOS*** results in the sparse surface with high approximation error.

Definition of CS^3: The performance analysis of various options leads to the conclusion that, ***CIOS*** is the most suitable for real-time applications. To summarize, CS^3 will include following steps: a pre-processing step of computation of a area-weighted normal map, minimum curvature-based importance computation, sampling based on importance-order which is followed by modified CDT triangulation for the visualization.

Table 3.3: Execution time analysis of different methods used for obtaining a sparse approximation of the surface of size 5% of original size. $*$ Underlined values indicate the best performance for corresponding categories.

	Time(ms)					
	QEM	**FMEPS**	**UNI**	**PDS**	**SSA**	**CS³**
Dragon	658	292309	**11**	95	28	**29**
Bunny	545	281311	**12**	107	29	**31**
Buddha	1109	550418	**14**	153	36	**45**
Desk	4247	1303350	**21**	149	N.A.	**62**
Human	4033	1354380	**20**	267	N.A.	**61**
Cabinet	4733	1069398	**25**	368	N.A.	**69**
Person 1	645	352262	**6**	109	17	**19**
Person 2	596	279197	**5**	99	16	**17**
Person 3	587	226344	**5**	78	15	**16**

Table 3.4: Approximation error analysis of different methods used for obtaining a sparse approximation of the surface of size 5% of original size. $*$ Underlined values indicate the best performance for corresponding categories.

	METRO$(*10^{-3})$						***MSE***$(*10^{-3})$						***FMPD***$(*10^{-1})$					
	QEM	**FMEPS**	**UNI**	**PDS**	**SSA**	**CS³**	**QEM**	**FMEPS**	**UNI**	**PDS**	**SSA**	**CS³**	**QEM**	**FMEPS**	**UNI**	**PDS**	**SSA**	**CS³**
Dragon	**0.334**	0.804	0.802	2.848	3.332	**0.694**	**0.15**	1.168	2.325	1.350	5.877	**0.713**	4.992	4.556	10.000	6.063	7.003	**4.033**
Bunny	**0.332**	0.743	0.767	0.675	2.547	**0.629**	**0.278**	0.293	4.318	1.393	2.915	**0.681**	2.679	2.266	3.851	5.170	10.010	**1.793**
Buddha	**0.406**	0.684	1.024	2.079	6.445	**0.663**	**0.385**	0.386	1.070	1.399	2.792	**0.675**	5.993	5.467	10.040	10.000	7.843	**4.422**
Desk	0.926	**0.766**	1.451	1.219	N.A.	**0.907**	**7.034**	9.387	10.417	9.861	N.A.	**8.389**	**1.346**	2.904	10.000	8.618	N.A.	**2.708**
Human	0.544	**0.515**	0.965	0.900	N.A.	**0.521**	**1.907**	2.255	3.087	2.135	N.A.	**2.015**	1.994	2.556	8.372	5.608	N.A.	**1.720**
Cabinet	0.795	**0.760**	1.191	0.918	N.A.	**0.768**	**0.584**	0.947	1.529	0.832	N.A.	**0.624**	**1.235**	1.887	8.292	6.892	N.A.	**1.431**
Person 1	**0.624**	1.003	0.846	0.795	2.316	**0.713**	**2.143**	8.075	16.615	6.581	32.122	**5.026**	4.979	3.907	5.646	5.701	6.263	**3.439**
Person 2	**0.594**	0.842	1.009	0.878	1.873	**0.736**	10.205	**5.051**	20.919	14.269	17.128	**6.685**	3.630	1.466	4.270	4.398	3.943	**1.220**
Person 3	**1.745**	1.848	2.351	1.907	4.698	**1.748**	**1.252**	2.682	6.768	6.012	8.878	**2.587**	2.692	1.516	3.431	4.884	4.603	**1.144**

3.6 Performance Comparison

We compare the performance of CS^3 with Uniform Sampling (UNI), Poisson Disk Sampling (PDS) (Hou et al., 2015), Scale-Space Analysis (SSA) (Cheng and Boulanger, 2006), Quadratic Error Metric (QEM) (Garland and Heckbert, 1997) based meshing and Fast and memory efficient polygonal simplification (FMEPS) (Lindstrom and Turk, 1998). We used a C++ implementation for both UNI and SSA. To apply QEM mesh simplification to the range data, first we compute a dense triangulation on the depth image as proposed in (Raghuraman et al., 2013a). On this dense mesh, QEM mesh simplification is applied using the open-source re-meshing tool provided by MeshLabv1.3.4 (Cignoni et al., 2008), which combines the *state-of-art QEM* method (Garland and Heckbert, 1997) with some post-simplification cleaning and refinement. Further, we used a quality threshold of 0.3 with boundary preserving weight set to a 1 so that faces with quality < 0.3 are penalized and the object boundary remains intact. PDS simplified surfaces are obtained by performing Poisson disk sampling using Meshlab tool followed by a constrained-Delaunay triangulation. For FMEPS, we utilized the implementation of polygonal simplification suggested by Lindstorm et al. (Lindstrom and Turk, 1998) in the open source computational geometry algorithms library (CGAL) (Cacciola, 2016). We assume that input depth and texture maps are pre-registered and pre-processed to remove the sensor noise.

Execution Time: Table 3.3 lists the execution time required by different methods. Uniform sampling needs minimum execution time whereas execution time for FEMPS and QEM mesh simplification is higher compared to other methods used for comparison. The proposed method can achieve much faster execution time compared to FEMPS and approximately 20 times faster than QEM mesh simplification. The time required for simplification of the ***complete 3D-scene data*** provided in CoRBS Dataset is slightly higher as the original dense models have more than 150K vertices. For visualization, the 3D data is rendered using tools such as OpenGL, Unity3D. The simplified data needs to be converted into data structure

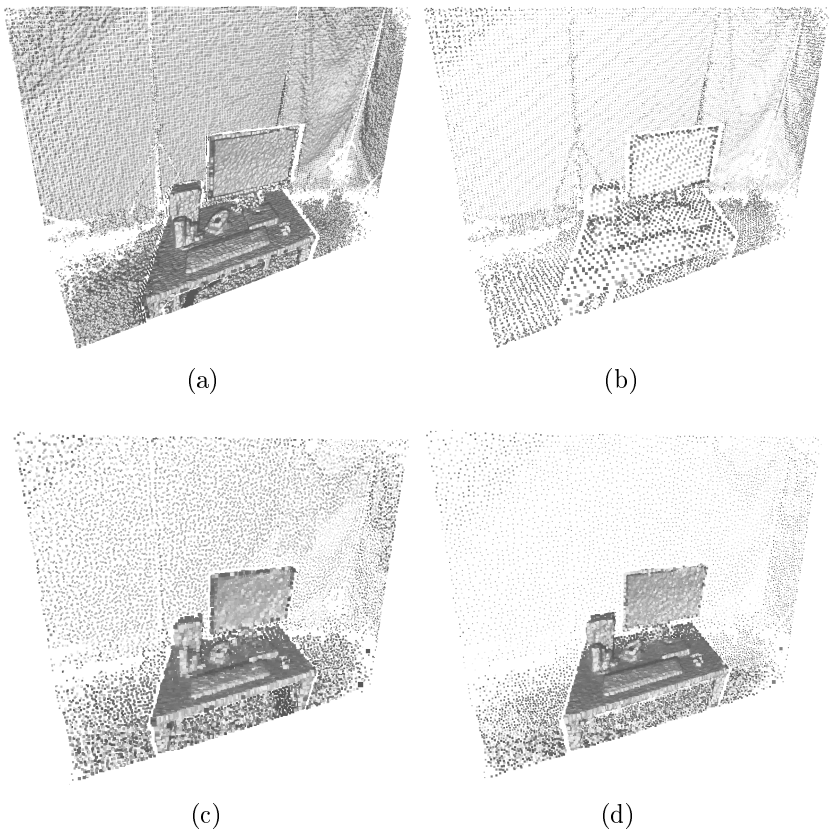


Figure 3.8: For the CoRBS desk model (Wasenmüller et al., 2016), a) Original dense set of points, Sampled at 10% of original size using b) uniform sampling, c) importance ordered based sampling and distance-dependent sampling.

compatible with such visualization tools. Experiments show that the time needed for such conversion is always less than 5 miliseconds across different data. It can be observed that the proposed method introduces very minimal latency that is not visually perceivable.

Note that the current implementation of the algorithm is on CPU. The execution time will be further accelerated on GPU-based implementation, especially because the computation of normal map and importance per point can be parallelized.

For applications such as 3D Tele-Immersion, user study done in (Wu et al., 2011) has shown that a frame rate of at least 10 fps is required for good immersive experience. As the average execution time for the proposed method is less than 100 ms, it is very much suitable for immersive applications.

Approximation to Surface: To test the ability of different methods to approximate the surface, we compute mean approximation error (METRO), fast mesh perceptual distance (FMPD) (Wang et al., 2012) and mean square error (MSE) for sparser representation of a surface obtained using the different methods. These quality metrics are widely used by prior researchers to quantify how well the sparse model approximates original surface (Garland and Heckbert, 1997; Hoppe, 1999; Wasenmüller et al., 2016). FMPD (Wang et al., 2012) computes the difference between the normalized surface integral of the local roughness (Wang et al., 2012). Mean square error is computed by computing the difference between original range image and the approximate range image obtained from a sparse mesh. Please refer to (Romanoni et al., 2016) and (Wang et al., 2012) for further details. Table 3.4 enlists the error induced by different methods for different models. It can be clearly seen that the proposed method provides an approximation error closest to the state-of-the-art QEM mesh simplification method. Figure 3.10 shows the visualization of METRO for the Stanford Dragon model. The approximation error is evaluated at every vertex of a dense mesh. Error distributions for different methods demonstrate that the proposed method induced less approximation error. Due to restrictive sampling, the proposed method generates sparse meshes with triangles which are more equilateral. Hence, it performs superior in terms of FMPD measure in most of the cases (see Table 3.4).

Preservation of Fine Details: As shown in Figure 3.9 and 3.12, the proposed approach can preserve finer details of surface even at very sparse levels. Facial features such as eyes, nose as well as wrinkles on cloth are well maintained. The proposed method is also able to maintain the integrity of the object shape unlike SSA as seen in Figure 3.11 f.

Preservation of Surface Attributes: Generally, the depth data captured using RGB-D cameras is accompanied with the color information that can be used to generate the textured 3D model. As the proposed method does not alter the position of points or do not add any additional points, it is possible to preserve the texture mapping. Generally, the

basic mesh simplification approach is modified to account for surface properties resulting into additional computation overhead and increased execution time. For example, Garland et al. extended the classic QEM mesh simplfication method (Garland and Heckbert, 1997) utilizing 4 x 4 quadric error to generalized QEM with 6 x 6 quadric error in order to handle surface properties in (Garland and Heckbert, 1998). For textured models, we also compare the proposed method with texture preserving QEM (Garland and Heckbert, 1998) and classic QEM (Garland and Heckbert, 1997) in terms of execution time and texture error as described in (Hoppe, 1999). The texture error is estimated by measuring the deviation of the texture coordinates of sampled points from the values linearly interpolated at their projection on the closest face on the sparse model. Figure 3.1, 3.13 and 3.14 illustrate that the method outlined in this paper does not show visible degradation in the textured rendering of the sparse surface approximation and achieves results similar to texture preserving QEM in significantly less execution time.

Elimination of Sensor Noise: Though, we assume that the input data is pre-processed to remove the sensor noise, the complete noise removal is a strenuous task. The proposed importance-order based restrictive sampling implicitly suppresses noisy points by virtue of repulsive force exerted by selected points. Additionally, the distance based triangle validation in the modified triangulation helps to further eliminate sensor noise as shown in Figure 3.12 and 3.13.

3.7 Extension for Distance-Dependent Simplification of the Complete Scene

Next, we consider a scenario where the entire scene is scanned using an RGB-D camera. In such a scenario, it is apparent that nearby objects grab most of the viewer's attention. Hence, it is crucial to approximate nearer objects more accurately compare to objects located at a distance. It motivates to design a distance-dependent sampling strategy that will densely sample the closer object while sparsely sampling far-away objects.

To capture this idea, we extend the importance-order based sampling proposed in Section 3.3.2 to inherently change the sampling density adaptive to object's distance from the viewer (i.e., the distance from an RGB-D camera). Using the depth measurement $R(\mathbf{u})$ at the sampled point and the maximum depth measurement R_{max}, we modify the definition of the corresponding radius of the affected region as:

$$Rd(\mathbf{u}) = \left(1 - \frac{\Omega(\mathbf{u}) - \Omega_{min}}{\Omega_{max} - \Omega_{min}}\right) * \left(\frac{R(\mathbf{u})}{R_{max}}\right) Rd_{max} \tag{3.9}$$

The modified definition of the affected region, allow the size of the affected region to be smaller for the sampled points of an object at a small distance. As the object's distance from the camera increases the area of the affected region due to selected points increases leading to more sparse sampling. Figure 3.8 illustrates the reduced set of points obtained by the uniform sampling, importance order-based sampling defining the affected region area as per equation and the proposed distance-dependent sampling strategy. The distance-dependent sampling outlined in this section helps to obtain the distance-dependent CS^3 (Curvature Sensitive Surface Simplification) keeping the framework of the original CS^3 unaltered.

Traditional quality metrics, such as METRO, MSE etc. measure the global error and do not account for the user's focus or proximity. Hence, the error analysis in terms of these quality metrics may not be appropriate for evaluating the distance-dependent approximation. The execution time analysis of the distance-dependent CS^3 shows that the time taken by distance-dependent CS^3 is similar to that of the original CS^3 indicating very minimal increased overhead.

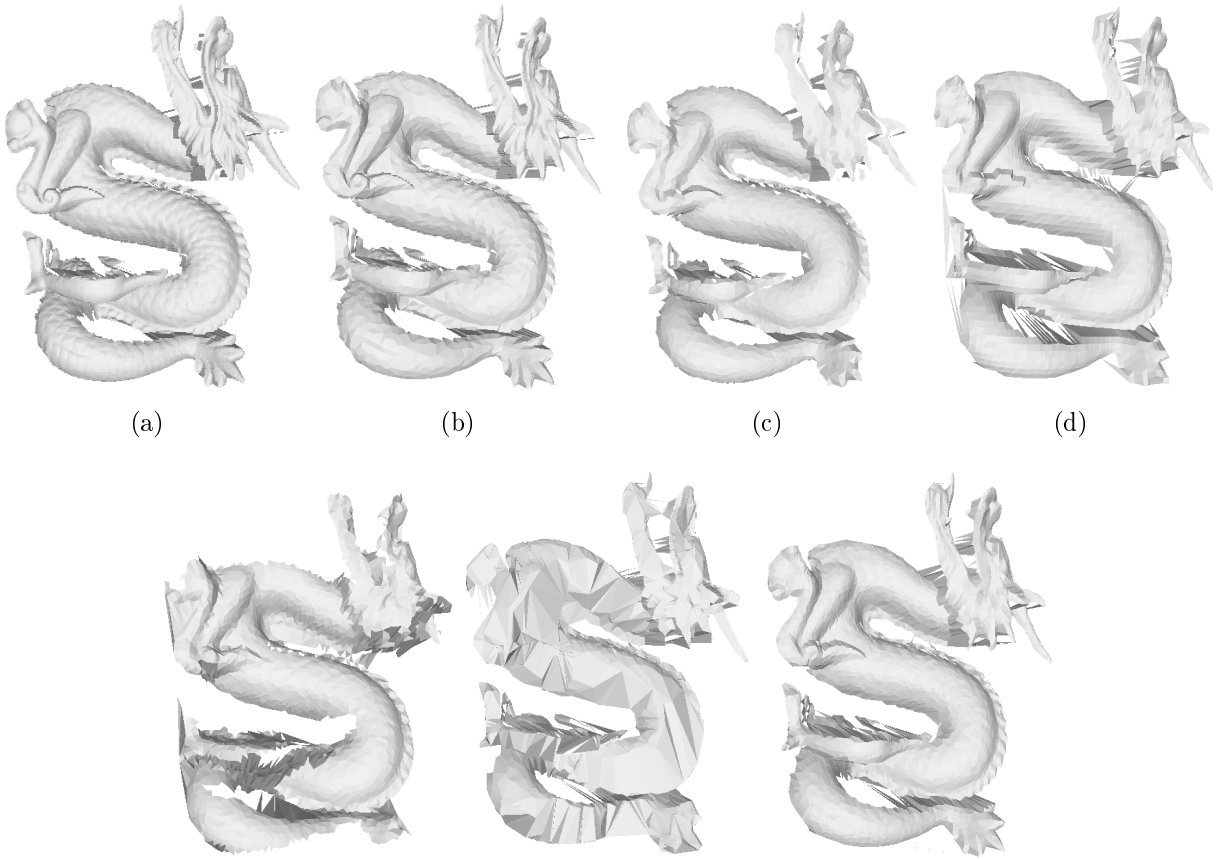


Figure 3.9: For the *Dragon* model from the Stanford Datset (Levoy et al., 2005), a) Original model, a sparser mesh with size 5% of the original model obtained using b) QEM (Qslim), c) FMEPS, d) UNI, e) PDS, f) SSA, and g) $\mathbf{CS^3}$.

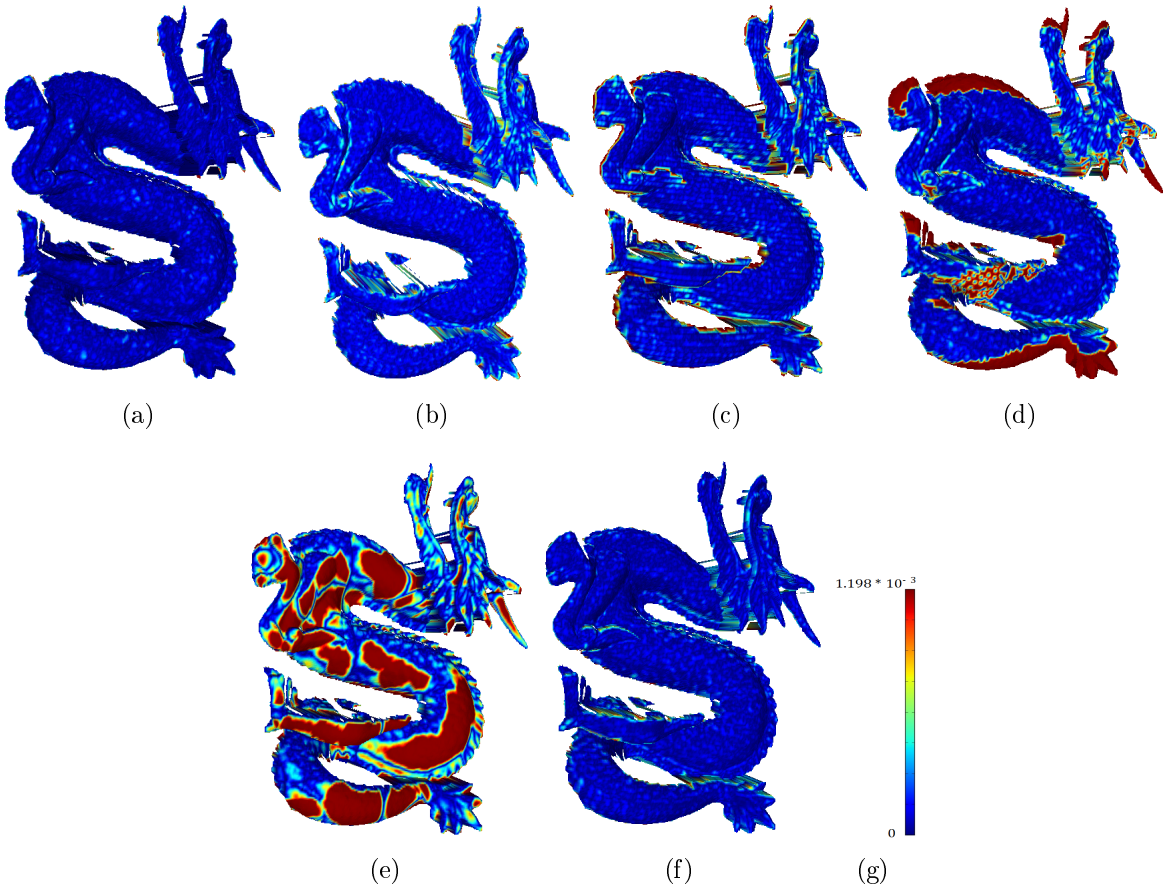


Figure 3.10: The visualization of METRO computed for a sparser mesh of *Dragon* from Stanford Dataset (Levoy et al., 2005) with size 5% of the original model obtained using a) QEM (Qslim), b) FMEPS, c) UNI, d) PDS, e) SSA, f) $\mathbf{CS^3}$, and g) the scale for METRO. Red indicates the high approximation error.

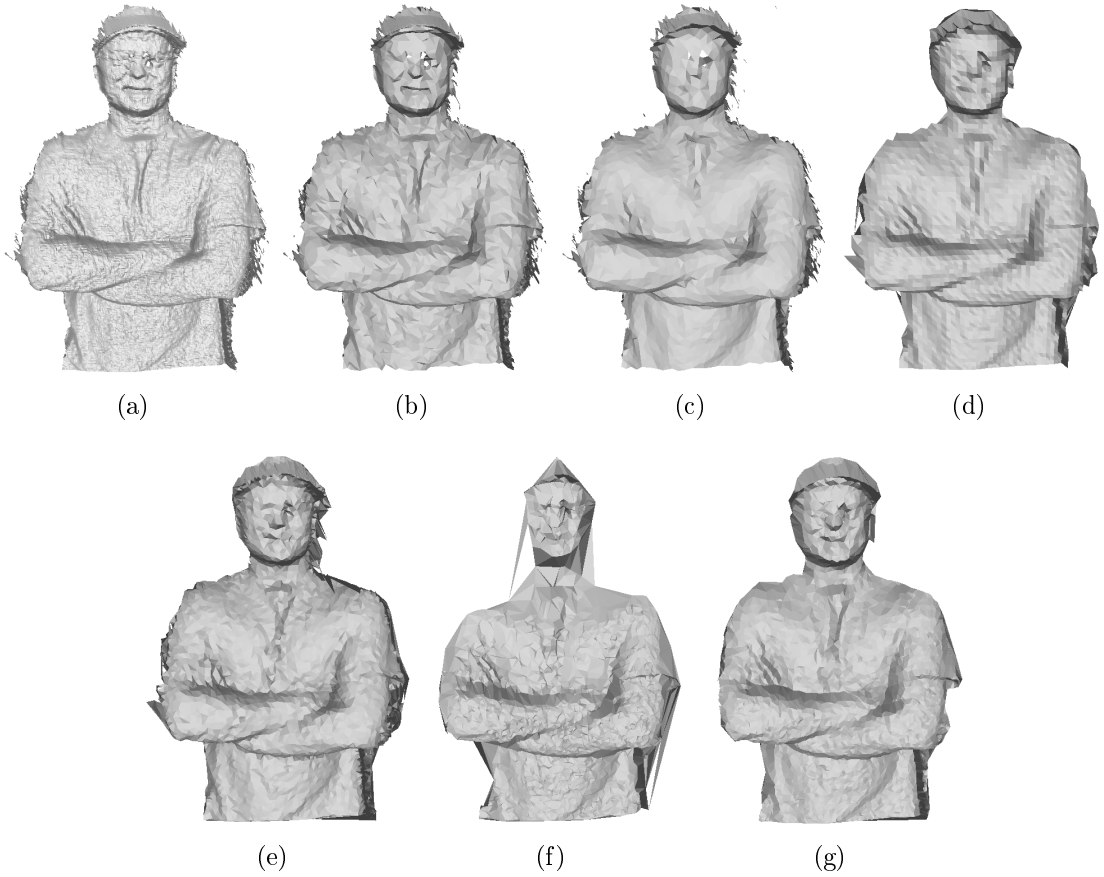


Figure 3.11: For the *Person 1* model, a) Original model, a sparser mesh with size 5% of the original model obtained using b) QEM (Qslim), c) FMEPS, d) UNI, e) PDS, f) SSA, and g) **CS^3**. Textured rendering is provided in the supplementary material.

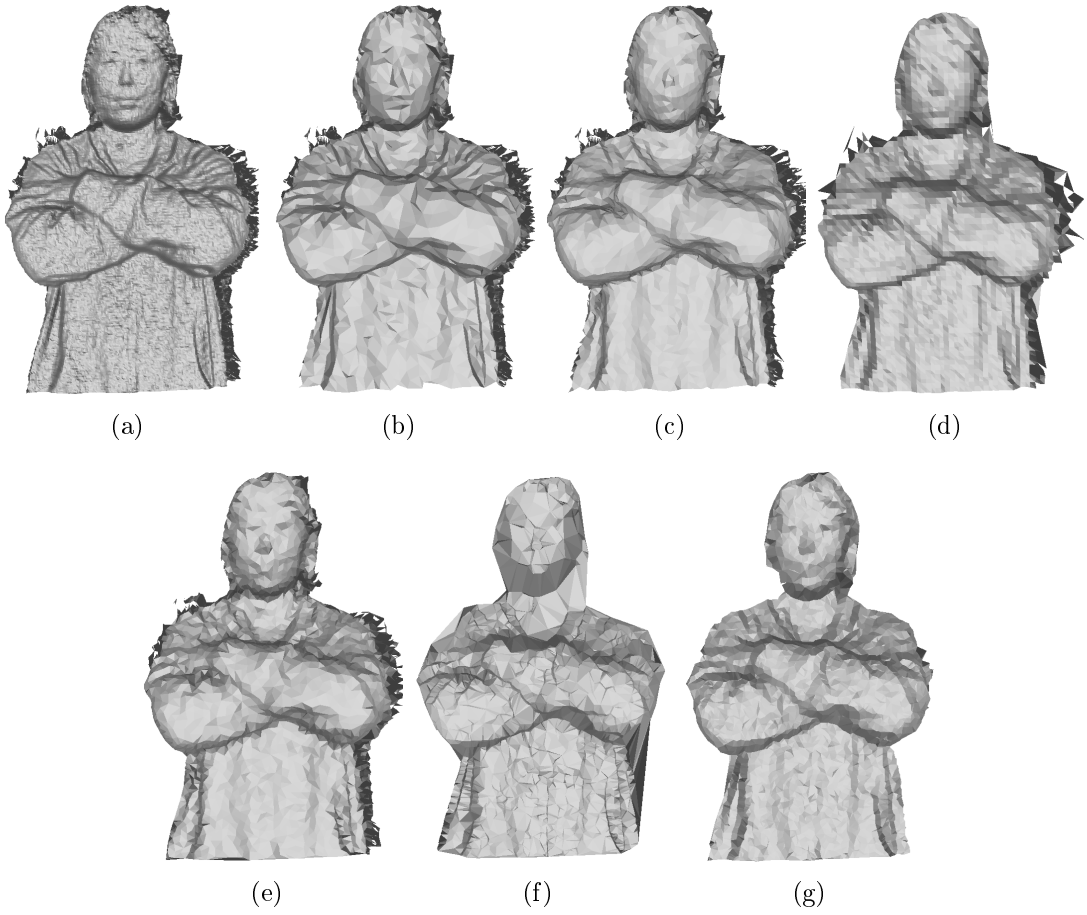
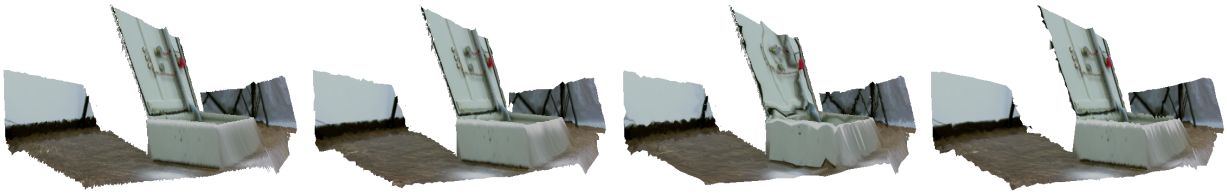


Figure 3.12: For the *Person 3* model, a) Original model, a sparser mesh with size 10% of the original model obtained using b) QEM (Qslim), c) FMEPS, d) UNI, e) PDS, f) SSA, and g) **CS^3**.

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

Figure 3.13: For the *Person 3* model, Textured rendering of a) the original model, a sparse mesh with size 10% of the original model obtained using b) texture-preserving QEM (texture error = $1.895{*}10^{-6}$, execution time = 736 ms), c) QEM (Qslim) (texture error = $66.05{*}10^{-6}$, execution time = 671 ms), and d) **CS³** (texture error = $3.584 * 10^{-6}$, execution time = 19 ms).



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

Figure 3.14: For the CoRBS *Cabinet* model (Wasenmüller et al., 2016), Textured rendering of a) the original model, a sparse mesh with size 5% of the original model obtained using b) texture-preserving QEM (texture error = $9.566 * 10^{-6}$, execution time = 4775 ms), c) QEM (Qslim) (texture error = $205.160 * 10^{-6}$, execution time = 4473 ms), and d) **CS³** (texture error = $12.562 * 10^{-6}$, execution time = 69 ms).

PART II

MODIFICATION FOR VIRTUAL THERAPIES

Virtual and Mixed reality have been successfully utilized for various virtual therapies. In this part, we explore the 3D content modification for creating an illusion of a phantom limb to alleviate phantom pain and also for providing positive reinforcement during virtual therapies.

Chapter 4: Mixed Reality for Managing Phantom Pain (Mr.MAPP) framework provides a virtual mirror therapy without the constraints of limited spcae, restricted movement and wavering illusion. It identifies and mirrors the intact limb in real-time to create the illusion of the phantom limb. Such an illusion provides a visual feedback for the phantom limb movement and aids in alleviating the phantom pain.

We also developed a virtual enhancement method that skeletal animation technique to enhance the movement of patient's live 3D avatar. Such virtual enhancement provides a positive reinforcement and assists in a rapid recovery in stroke patients.

CHAPTER 4

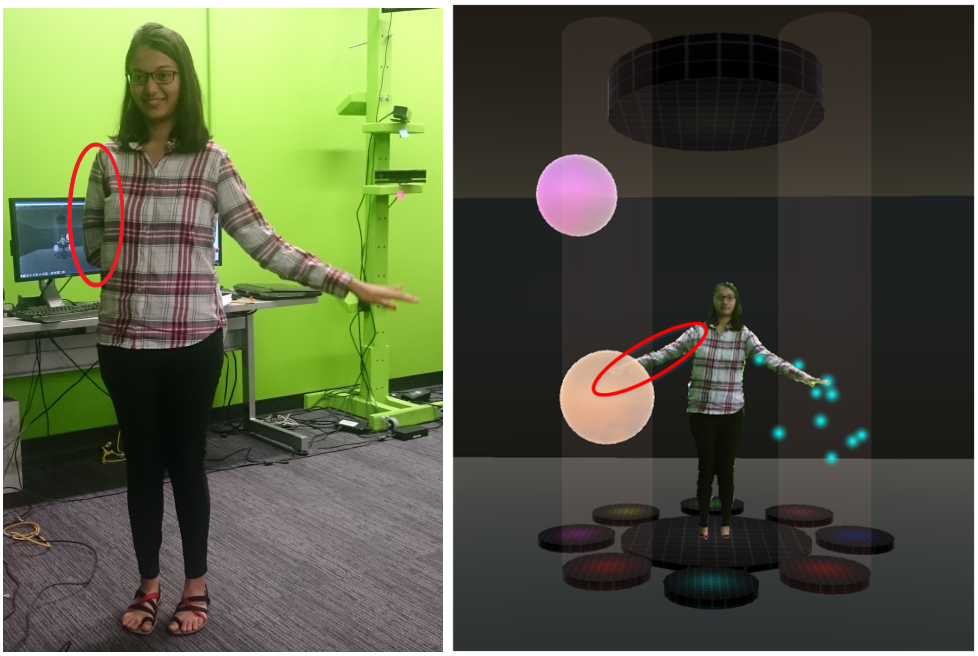
MR.MAPP: MIXED REALITY FOR MANAGING PHANTOM PAIN [1]

4.1 Introduction

Subsequent to amputation, a patient commonly experiences the residual sensation of the limb creating the illusion that their missing limb is still intact. This residual sensation of the missing limb is often referred as a phantom limb. Along with the phantom limb, patients commonly develop a painful sensation that is perceived as stemming from the missing limb (i.e., phantom limb pain). The Phantom Limb Pain (PLP) is a chronic pain which is considered as one of the most traumatic consequences of amputation. Research has shown that the phantom limb pain is a serious cause of severe distress and physical limitations in 85% of amputees (Sherman et al., 1984; Murray et al., 2009). Restriction in normal activities and higher levels of depression often severely affects patient's social and work life. Although various surgical, psychological and pharmaceutical methods are employed to treat PLP, the effectiveness of these methods is often limited and short-term (Katz, 1992).

Flor et al. (Flor et al., 1995) showed that phantom limb pain is closely related to neuroplastic changes in at least the primary somatosensory cortex. Based on neuroplasticity, Ramachandran et al. (Ramachandran and Rogers-Ramachandran, 1996) designed a classic method of mirror therapy for the relief of the phantom limb pain. In mirror therapy, a vertical mirror is placed inside a cardboard box with the top removed. The patient is asked to place his/her intact limb in the box such that its reflection in the mirror gets superimposed on the felt position of the phantom limb. Various studies have shown that the visual clue provided by the mirror therapy can induce the vivid sensation of the movement stemming from the muscles and joints of the patient's limb. Blakemore et al. (Blakemore et al., 2002) have aptly

[1]



(a) (b)

Figure 4.1: a) Real scene with healthy user keeping her right hand behind and b) Virtual scene with phantom limb.

explained the fundamental principle of mirror box therapy in terms of the internal forward model of the central nervous system. Whenever any movements are performed, the forward model predicts the sensory consequences of motor commands. Hence, the experience of the limb movement is generally based on the predicted rather than the actual state. But when the limb is missing, motor commands are still issued and the movement is predicted by the forward model simulating the experience of the movement of the phantom limb. However, as the limb is not actually moving, there is a discrepancy between the predicted movement and the visual feedback of the actual state. The mirror box therapy allows to complete this brain circuitry by providing the visual clue for the phantom limb movement and can restore the movement of the phantom limb voluntary.

However, the mirror box therapy is highly constrained by the limited spatial movement, requirement of a patient to remain in the fixed posture. Furthermore, the illusion obtained

is wavering in nature and requires the patient to pay continual attention only to the reflected image that can be tedious and stressful. To overcome the challenges presented by the mirror box therapy, various virtual reality-based methods are suggested in the literature. Most of these methods utilize a pre-built 3D model of the phantom limb generated using graphics tools. The movement of amputee's intact limb is captured using various body sensors and transposed to the 3D model of the phantom limb. Virtual reality-based methods provide the similar illusion to the mirror box therapy while providing the higher flexibility in the movement and without the space constraints. However, these methods are highly susceptible to degraded immersive experience due to a mismatch in the skin color, clothes, artificial and rigid look and misalignment of the phantom limb. Further, most of the virtual reality-based methods proposed in the literature rely on external devices or sensors to be worn by a person to capture the motion. Wearing such sensors can inhibit the natural user movement and can also cause skin irritations and discomfort (Brütsch et al., 2010; Chen et al., 2011, 2010; Duff et al., 2010; Vieira et al., 2015). Due to the usage of such body sensors, these methods can be considered as invasive and presents difficulties in employing them in pain relief sessions.

4.1.1 Proposed framework for Managing Phantom Pain

Motivation: With the recent advances in the depth sensing technology and wide availability of low-cost depth sensors, it has become feasible to capture a *live* 3D model and motion data of the person that facilitates the user to interact with virtual objects with complete immersive experience. Using the powerful tool of active depth sensing, a live 3D model of the phantom limb can be generated in the real-time instead of using the rigid and artificial looking pre-built models of the limb. Also, the motion data provided by depth sensors allows designing a non-invasive approach that does not require additional body sensors for capturing the motion data.

To address the issues mentioned in the section 4.1, we propose a novel Mixed Reality based system for MAnaging Phantom Pain (Mr.MAPP). The proposed framework employs off-the-shelf RGB-D cameras such as Microsoft Kinect V2 (Zhang, 2012) to capture and generate a 3D model of the person in real-time. An illusion of the virtual limb is crafted by mirroring the patient's symmetric anatomical limb in the captured data using various computer vision and graphics techniques. The major challenges in developing the Mr.MAPP framework are: a) capturing the live 3D models of the human, b) generating the 3D model for the phantom limb in the real-time, c) generating corresponding movement in the real-time, and d) rendering the 3D model of the person along with phantom limb for the immersive and interactive experience. Also, to make the therapy sessions more engaging, fun and non-monotonous, it is also required to design an attractive virtual environment and game with the engaging task. Figure 4.1 shows the virtual environment developed for Mr.MAPP framework.

In this paper, we also present a comparative study to analyze the effect of various rendering displays such as 3D Television, and Head mounted displays (Oculus Rift, Samsung Gear VR) in the overall quality of experience. To determine the effectiveness of the proposed system, Mr.MAPP framework is evaluated in terms of its usability and acceptability by clinicians and rehabilitation physicians.

4.1.2 Principal Contributions

The proposed Mr.MAPP framework was tested and evaluated by total of 27 users broadly categorized into two classes: a) 11 Subject-Matter Experts that includes 5 Physical Medicine and Rehab (PM&R) experts, 3 Amputee Occupational Therapist and 3 Doctors of Chiropractic, and b) 16 people with no known disabilities. The principal contributions of this paper include:

- A real-time identification and removal of points belonging to the missing limb if there are any.

- A real-time phantom limb generation by mirroring the intact limb.
- A real-time skeleton generation for the phantom limb to allow interactions with the phantom limb in the virtual world.
- A study to evaluate the effectiveness of the various display methods in creating an immersive experience and consequently in pain management.
- A detailed evaluation of the system from Physical medicine and rehab physicians and psychology experts for validating the efficacy of the proposed Mr.MAPP framework and its ability to provide interactive, engaging, motivating and enjoyable system for managing the phantom limb pain.

The work described in this paper has following impacts:

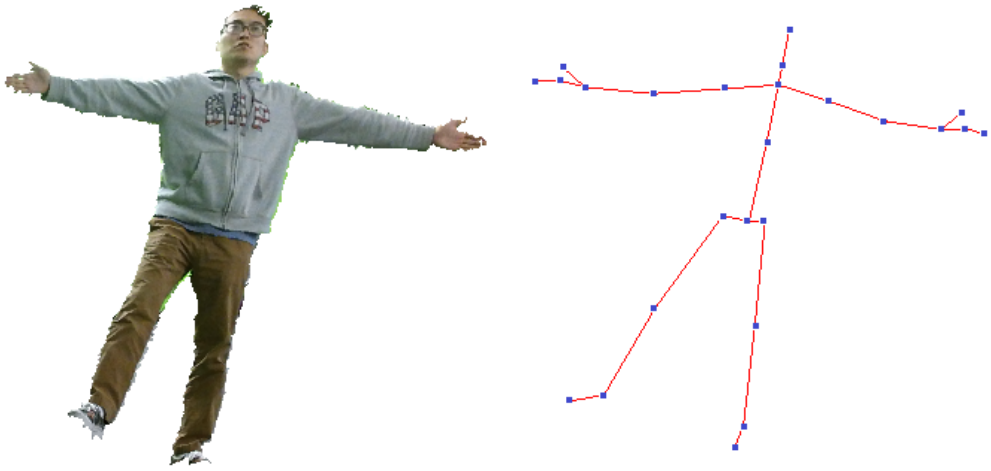
- A novel Mixed Reality based system for MAnaging Phantom Pain (Mr.MAPP) which creates a phantom limb using augmented virtuality. It provides a cost-effective solution that is simple and easy to use for the relief from phantom limb pain.
- More realistic and natural representation of the phantom limb that matches with the person's skin tone and clothing.
- A non-invasive approach to obtain the virtual movement of the phantom limb.
- Interaction-enabled, lifelike, and smooth movement of the phantom limb.

4.2 Related Work

Generally, different surgical, pharmaceutical and psychological interventions are used for treating the phantom limb pain. However, these methods have very limited and short term success (Katz, 1992).

One of the promising methods, a mirror-box therapy was proposed by Ramachandran et al. (Ramachandran and Rogers-Ramachandran, 1996) that completes the brain circuitry in the forward model of the central nervous system. It provides the visual feedback of the phantom limb movement using the mirror-box and can provide the relief from a phantom limb pain. Research studies done by Brodies et al. (Brodie et al., 2003) and MacLachlan et al. (MacLachlan et al., 2004) demonstrate that the mirror box therapy can successfully reduce PLP in a lower-limb amputee as well.

Recently, researchers have developed virtual reality-based methods that provide the similar illusion to the mirror box for treating phantom limb pain. O'Neil et al. (O'Neill et al., 2003) have designed a virtual mirror box that uses a graphical 3-D representation of the arm. The virtual 3-D arm is controlled by a wireless glove and visualized on the flat computer screen. The participant is asked to wear a wireless glove on their intact arm. Using the sensor embedded in gloves, the movement of the intact arm is applied on the phantom arm to move it in unison. Using the same virtual mirror box, Desmond et al. (Desmond et al., 2006) have presented three case studies indicating it's potential to treat phantom pain. Cole and colleagues(Cole, 2008)(Cole et al., 2009) have developed a virtual system that utilizes the remaining portion of an amputated limb to control the phantom limb. The controlling movement is obtained by placing the motion capture device on the remaining part of the amputated limb. As the motion capture device provided the overall movement, fine movements such as finger movement are pre-animated and not acquired in real-time. In contrast to these two approaches, Murray and colleagues have developed a system where a head-mounted display is used to visualize the virtual environment instead of a flat screen. The body sensors are attached to shoulder, elbow and wrist joint and thigh, knee and ankle joint for upper and lower limb amputees respectively. Motion obtained via body sensors is then applied to the pre-built 3D model of the phantom limb. Ortiz-Catalan et al. (Ortiz-Catalan et al., 2014)(Ortiz-Catalan et al., 2016) proposed a myoelectrically controlled AR



(a) (b)

Figure 4.2: a) Textured mesh and b) skeleton obtained from the data captured using Microsoft Kinect V2.

environment (MCARE) where a webcam is used to capture the whole environment around the person which is augmented with the 3D model of the phantom limb. Eight bipolar electrodes and markers are placed around the stump to obtain EMG signals. Using the recorded EMG signals, Linear Discriminant Analysis in a one-vs-one topology and multi-layer perceptron are trained for developing a myoelectric pattern recognition that predicts intended motion.

All of these methods rely on the pre-built 3D model of the phantom limb. The usage of such pre-built 3D model significantly degrades the quality of immersive experience due to a mismatch in skin tone, clothes, misalignment and artificial and rigid look of the phantom limb. Further, the pre-built model needs to be modified such that its dimension are in proportion with the current participant's anatomy. Most of these methods require the participant to wear data gloves, body sensors or electrodes to capture the motion data. Wearing such sensors not only hinders the natural movement but can also cause skin irritations and discomfort. We address these issues with the presentation of Mr.MAPP framework.

4.3 Mixed Reality for MAnaging Phantom Pain (Mr.MAPP)

Visual feedback can potentially induce the illusion of the restored limb and allow to perform phantom movements. To provide such realistic visual cues, we propose a novel mixed reality based system. It entails capturing and generating a 3D mesh model of the participant in real-time and placing it in the virtual world. It is also required to estimate the human skeleton to facilitate the natural interaction with the virtual objects.

A real-time 3D model of the participant is generated using off-the-shelf, RGB-D sensors such as Microsoft Kinect V2. Microsoft Kinect V2 provides two raw data streams: depth data and color data. The raw depth data is processed to filter out the human from the entire scanned scene by applying a depth filter that masks the background and generates depth bounds for the foreground (i.e., Human in this case) (Raghuraman et al., 2015a). Using the segmented depth data and camera intrinsic parameters, a 3D point cloud corresponding to the human is obtained by back-projecting each depth point into a real world. A 3D point cloud and the raw color data are combined together to generate a textured mesh corresponding to the human using a dense mesh strategy described in (Pajarola et al., 2003). This method considers a 2x2 neighborhood around each depth pixels and checks the validity of each possible triangle between these four pixels. The validity of a triangle is determined based on the depth variation in the points forming the triangle. The segmented depth streams are further processed to extract features that along with random decision forest algorithm are used to estimate human pose, as described in (Shotton et al., 2013). A human pose is represented as a skeletal model with 25 joint position and corresponding orientations. Figure 4.2 shows the example of the textured mesh and corresponding skeleton.

4.3.1 Real-time Phantom Limb generation

Though the above-described procedure provides a real-time 3D model of the participant, the 3D model corresponding to affected limb is still missing. To create a visual and interactive representation of the phantom limb, we need to generate:

- 3D mesh corresponding to phantom limb for visualization.
- 3D skeleton corresponding to phantom limb for performing interactions with the virtual objects.

Real-time Phantom 3D Mesh Generation

Getting inspired from the classical mirror-box therapy, we propose a phantom 3D mesh generation scheme that exploits the inherent symmetry in the human anatomy. It generates the 3D model for the affected limb by mirroring the 3D mesh of participant's intact limb. The proposed 3D mesh generation scheme has following steps:

- Point cloud segmentation
- Removal of points belonging to the affected limb
- Mirroring the intact limb

Point cloud segmentation: To identify the points corresponding to the intact limb, we establish the correspondence between the scanned point cloud and each skeletal segment. The correspondence is established by performing the skeletal segmentation of the scanned point cloud as described in the (Raghuraman et al., 2013b). The entire point cloud is segmented into a various segmented region based on the Euclidean distance between the point and the skeletal segment. Each skeletal segment consists of a control point $P_c = [x_c, y_c, z_c]^T$ and a reference point $P_r = [x_r, y_r, z_r]^T$. Generally, the corresponding skeletal joint is considered as

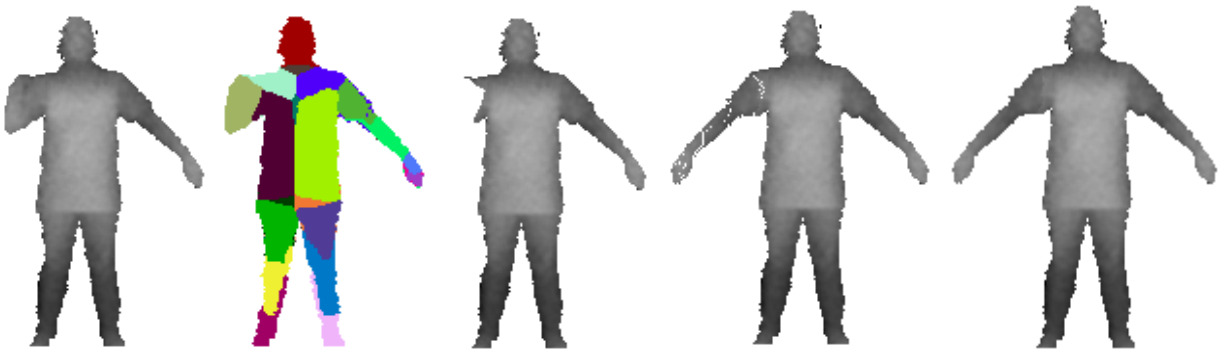
a control point, and the control point of the next skeletal segment is considered as a reference point for the current segment. A Voronoi decomposition based approach (Aurenhammer, 1991) is used that estimates the distance between a point and the skeletal line segment defined as (P_c, P_r). For a point $P(x, y, z)$ and line segment (P_c, P_r), the distance d is given as:

$$
\begin{aligned}
d &= \begin{cases} |P - P_r| & \text{if } t \geq 1 \\ |P - P_c| & \text{if } t \leq 0 \\ \frac{|(P-P_c)\times(P-P_r)|}{|P_r - P_c|} & \text{if } 0 < t < 1 \end{cases} \\
t &= \frac{(P - P_c)^T.(P_r - P_c)}{|P_r - P_c|^2}
\end{aligned}
\tag{4.1}
$$

where t represents the projection of the point P on the line segment (P_c, P_r). Figure 4.3 b shows the segmented point cloud.

Removal of points belonging to the affected limb: To achieve a realistic illusion, the points corresponding to the affected limb in the current scanned data needs to be removed. To identify all the points belonging to the affected limb and to label them with a single unique label, we modify the segmentation described above. For the affected limb, we collapse the local bone hierarchy (such as shoulder $\rightarrow$ *elbow* $\rightarrow$ *wrist* $\rightarrow$ *hand*) to a single bone from shoulder to hand. After identifying all the points corresponding to affected limb with a unique label, we remove these points from the depth image by assigning the zero depth value to corresponding pixels in the depth image. Figure 4.3 a and c show the original raw depth map and shows the depth map after removing the points corresponding to the affected limb.

Mirroring the intact limb: A 3D virtual representation of the phantom limb is crafted by mirroring the 3D model of the intact limb. Points corresponding to each skeletal bone of the intact limb are identified using the skeletal segmentation as described earlier. After segmenting the data, it is easier to mirror each individual skeletal segmented region to obtain its symmetric counterpart.



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

Figure 4.3: a) Original depth image, b) segmented point cloud, c) modified depth image after removing points corresponding to the right arm, d) modified depth image after adding points belonging to the phantom limb obtain by mirroring the left arm, e)depth image after applying the hole filling.

The skeletal segmented regions are mirrored by aligning the axis of symmetry with the medial axis of the participant. To achieve this, in the case of the upper limb amputee, we define a local coordinate system of participant's skeleton with the origin defined at the *spine joint* P_{sp} as:

$$x_s = \frac{P_{ls} - P_{rs}}{|P_{ls} - P_{rs}|}, y_s = \frac{P_{sc} - P_{sp}}{|P_{sc} - P_{sp}|}, z_s = \frac{x_s \times y_s}{|x_s \times y_s|}, O_s = P_{sp} \tag{4.2}$$

where P_{ls} and P_{rs} are left and right shoulder joint, P_{sp} is spine joint, P_{sc} is a shoulder center joint.

For mirroring the points, each point P_w of the intact limb is first transformed to the local coordinate system of the skeleton, reflected about the symmetric axis and then again transformed back to the real world coordinate system to obtain the mirrored point representing

the phantom limb. It is mathematically described as:

$$P_{t1} = P_w - O_s \tag{4.3}$$

$$P_s = M * P_{t1} \tag{4.4}$$

$$P_r.x = -P_s.x, P_r.y = P_s.y, P_r.z = P_s.z \tag{4.5}$$

$$P_{t2} = M^{-1} * P_r \tag{4.6}$$

$$P_{rw} = P_{t2} + O_s \tag{4.7}$$

where $M = [{x_s}^T {y_s}^T {z_s}^T]$ defines the rotation between skeleton local coordinate system CS_{sk} and real world coordinate system CS_{rw}, P_s is transformed point in the CS_{sk}, P_r and P_{rw} are reflected points in the CS_{sk} and CS_{rw}, P_{t1} and P_{t2} are intermediate results. We add mirrored points to the depth image by assigning it's z coordinate value to the corresponding pixel in the depth image. Figure 4.3 d shows the depth map after adding the points corresponding to the phantom limb.

For the newly added points, the traditional approach of generating textured mesh using the calibration parameters between depth and color camera will generate the incorrectly textured mesh. In Figure 4.4,the left model shows the example of such a textured mesh generated with the traditional texture mapping. To alleviate this issue, for each mirrored point P_{rw} in the depth image, we maintain the index of the corresponding original point P_w. The index idx of any point $p(i, j)$ in the depth image with width wh is defined as: $idx = i*wh+j$. Maintaining the index in this manner allows to easily trace back the original point. Hence, for the mirrored points, instead of computing the texture mapping directly from the calibration parameters, the original point can be referred using the associated index and corresponding texture mapping can be extracted. In Figure 4.4, the right model shows the textured mesh generated for the added phantom limb using the index based texture mapping approach.

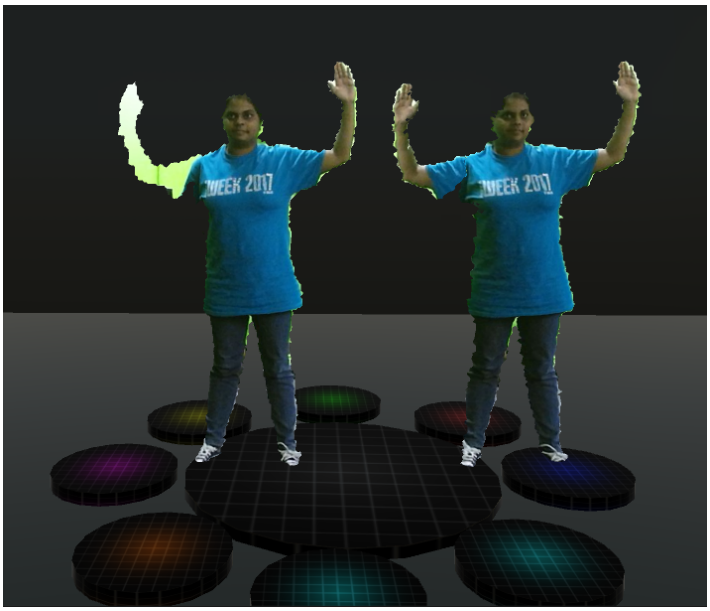


Figure 4.4: Textured mesh generated from Point cloud data using traditional texture mapping (left) and textured mesh generated from the same point cloud with our proposed approach.

Successive projection and back projection of points induce numerical round-off errors that lead to loss of few points and results into holes in the depth image as shown in Figure 4.3 d. To fill these holes, for each pixel with zero depth, we apply a conventional average filter to its 4-connected neighborhood. Further, the reference index for zero depth pixel is obtained by interpolating indices of it's immediate two horizontal neighbors or vertical neighbors with non-zero depth values. Figure 4.3 e shows the depth image obtained after the proposed hole filling step.

Real-time Phantom Skeleton Generation

For the immersive and interactive experience, it is important to provide the full body interaction using physics colliders. In the proposed framework, as the new mesh is created every frame, estimating the mesh collider every frame adds a significant latency to the system performance. Hence, physics colliders are estimated using the skeleton. Each skeletal bone

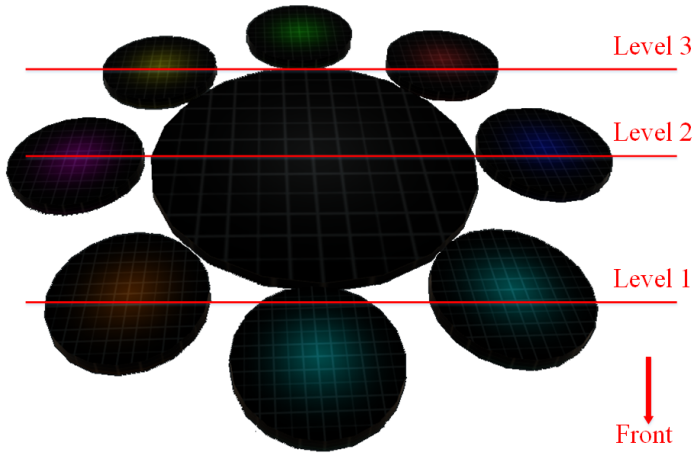


Figure 4.5: Layout for Bubble generation.

segment is associated with either a box or a capsule collider to cover the complete mesh volume.

Microsoft Kinect V2 provides an estimated skeleton per frame. Although it provides an overall good skeleton estimate, it can not estimate skeletal information for the missing limb. We obtain the skeletal information for the missing limb by mirroring the skeleton information of the intact limb obtained from Microsoft Kinect V2. We apply equations 3-9 to each skeletal joint of the intact limb to obtain the mirrored joint representing the phantom limb. Further, the local bone hierarchy for the phantom limb is maintained similar to that of the intact limb.

4.4 Immersive Game using Mr.MAPP Framework

To make the training session more attractive and engaging, we have designed an interactive game *Bubble Burst*. In *Bubble Burst*, the bubbles filled with different color paints are popped from the generators located on the floor and are floating towards the ceiling in straight lines. The participant is supposed to pop the bubbles in order to save the ceiling from getting stained with the paint. The game is designed while keeping in mind, the requirement of the symmetric and synchronized movement of a person's both arms. For realistic illusion

and accurate visual cues, it is important that the movement of the phantom limb should be in accord with the targeted movement, i.e., if bubbles from both sides are popping up in synchronization then the symmetric movement of the intact limb and phantom limb will be more realistic and conceivable. Hence, to enforce the synchronized movement of both limbs, we pop the bubble from left and right side in synchronized manner.

In this game, the person is surrounded by bubble generators. It has three different levels for the position of bubble generation. At any point in time, bubbles are popped out from two bubble generators at the same level in a synchronized manner. Figure 4.5 shows the top view of the virtual scene in the game describing the layout for bubble generation. Hence, the bubble generators at level 1 pops out bubbles at the same time. *Bubble Burst* has multiple levels of difficulty. The difficulty levels are controlled by changing the speed and location of popping bubbles.

4.5 Experimental Setup

The entire experimental setup involves only a single desktop with sufficient processing power, a single Microsoft Kinect V2 (Zhang, 2012) for capturing the 3D mesh data, and one rendering display for visualization. In our experiments, we have used a computer with a GTX 970 graphics card, Intel i7 2.4 GHz processor and 32 GB RAM. We employed a single Microsoft Kinect V2 connected to the computer using a USB 3.0 to capture and extract the person present inside the scene. Using the captured data, a textured 3D model is generated as described in the section 4.3.1. The generated textured mesh is immersed in a virtual game environment for interactive experience. The entire motion capture and Mr.MAPP framework is written in C++, and the game development is done in the Unity3D game engine with C# platform using OpenGL rendering and PhysX for physics. TCP sockets were used to communicate between the Mr.MAPP framework and the *Bubble Burst* game in Unity3D.

For visualization, the virtual game and the immersed 3D model of person are rendered on various 3D displays.

To analyze the effect of various 3D display technologies on the immersive experience provided by Mr.MAPP framework, we have experimented with three different displays such as: A Samsung 3D TV, Oculus Rift, and Samsung Gear VR. A Samsung 3D TV provides a partial 3D immersive experience without constraining the user movement. On the other hand, head mounted display HMD such as Oculus Rift, and Samsung Gear VR provides a complete 3D immersive experience. Among these two HMDs, the Oculus Rift is connected to the desktop, whereas Samsung Gear VR relies on a mobile device to provide portability and ease of operation. Due to their unique features, these three displays are considered for experiments. In case of 3D TV and Oculus Rift, both these devices are connected the desktop that runs the Mr.MAPP and game application. Whereas in case of Gear VR, the game application is run on the mobile device and the 3D model of person is transmitted from the desktop to the mobile device via a WiFi adapter. For Gear VR, we have used Samsung S6 edge plus as a mobile device.

4.5.1 Evaluation of the Mr.MAPP framework

We evaluated the Mr.MAPP framework in terms of three key factors: a) visual quality, b) real-time performance and c) user experience.

Visual Analysis:

We demonstrate the efficacy of the proposed Mr.MAPP framework by providing the visual rendering of the Phantom Limb created using it. Figure 4.4 and 4.6 show the examples of the phantom limb generation in case of the upper limb and lower limb amputation respectively. For simulating the lower limb amputee, the participant was asked to sit on the chair with folding one of his/her leg as shown in Figure 4.6 a. Figure 4.6 c shows the textured model

(a) (b) (c)

Figure 4.6: a) Real scene with the black box in front of the person, b) textured mesh captured from Kinect, and c) textured mesh after adding phantom lower limb.

with phantom lower limb. Additional videos showcasing the upper and lower phantom limb generation are provided in the ***supplementary material***.

Real-time Performance Analysis:

To demonstrate the real-time performance of the Mr.MAPP framework, we carried out various experiments. Due to perspective projection based depth sensing, the number of points representing the 3D model of a person varies based on its distance from the sensor. Consequently, the execution time is highly influenced by the distance of the person from the sensor. Hence, to account for this distance dependency, we carried out a distance based time performance analysis.

During different stages of the experiment, a person was asked to stand at the distance of 8, 10 and 12 feet from the sensor in the same pose. For each stage, the time taken by each step of the Mr.MAPP framework is measured and logged. Table 4.1 shows the time measurements for different steps involved in Mr.MAPP at various distances. It can be

Table 4.1: Execution time.

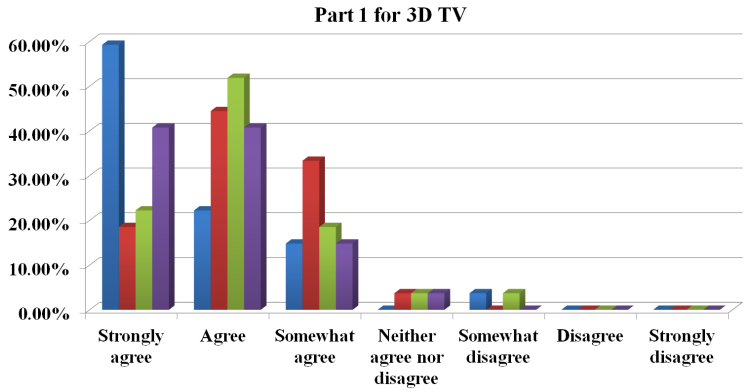
Process	**Time Taken** *(in milliseconds)*		
	at 8 feet	*at 10 feet*	*at 12 feet*
Point cloud segmentation	4.76	3.1543	2.2169
Removal of points	0.3357	0.2857	0.240
Mirroring the intact limb	0.882	0.833	0.783
hole filling	0.0004	0.0004	0.0003
Phantom skeleton generation	0.0001	0.0009	0.0009
Point cloud generation	1.425	1.278	1.1863
Textured mesh generation	3.6348	2.7742	2.2981
Total Time	**11.033**	**8.32569**	**6.72469**

seen that the process of phantom limb generation approximately adds only 6 milliseconds of overload. The overall execution time of the Mr.MAPP framework is always less than 11 milliseconds at various distances. Thus, the Mr.MAPP framework guarantees the real-time phantom limb generation.

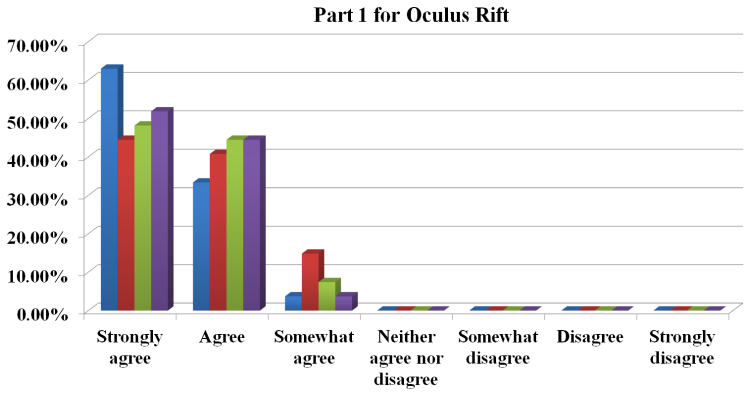
User Study:

To perform a subjective analysis of the Mr. MAPP framework and to understand the effect of various 3D display technologies on the immersive experience, we performed an user study with total of 27 participants. It involved two classes of users: a) a group of 11 Subject-Matter Experts in the area of amputee rehabilitation that includes 5 Physical Medicine and Rehab (PM&R) experts, 3 Amputee Occupational Therapist and 3 Doctors of Chiropractic, and b) 16 healthy adults with no known disabilities. 18 of the participants were male, and 9 were female. Age of the participating people was from 23 to 55.

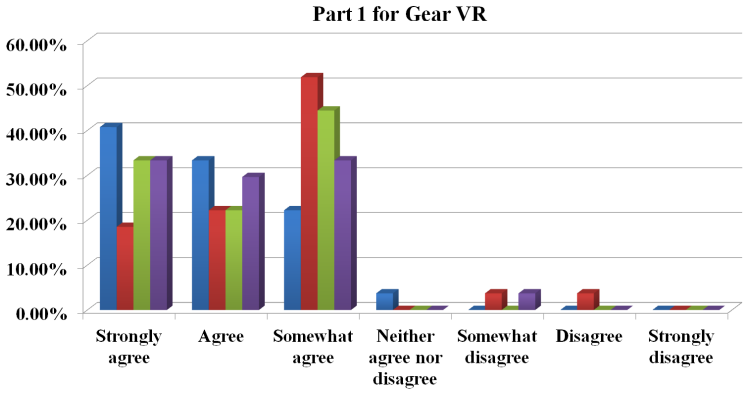
At the beginning of the study, we explained the task require to be performed to each participant by giving a brief demonstration of the game. Each participant is also asked to move and explore the virtual game environment for a time duration of 3 minutes in order



(a)

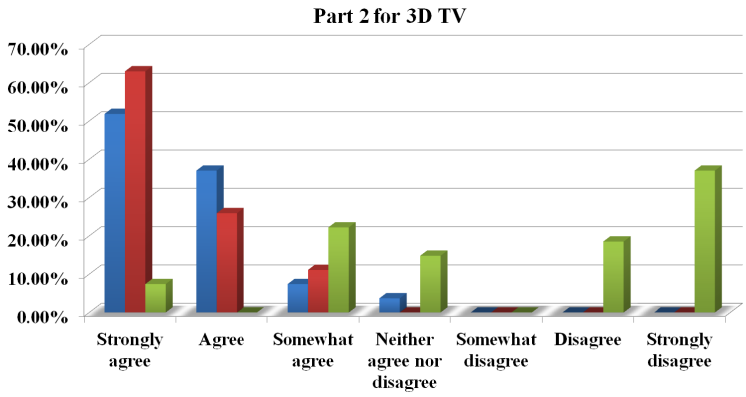


(b)

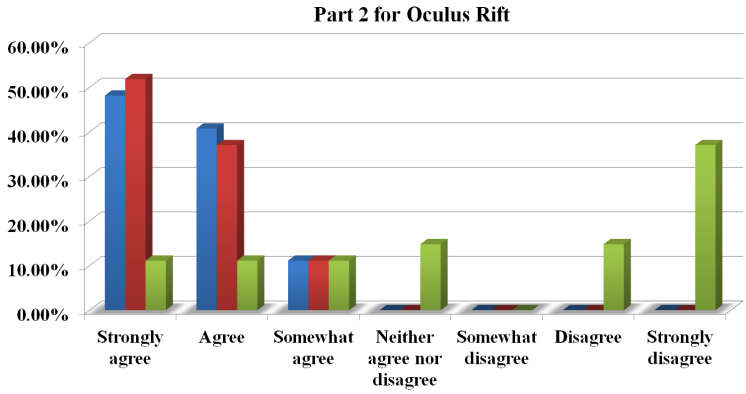


(c)

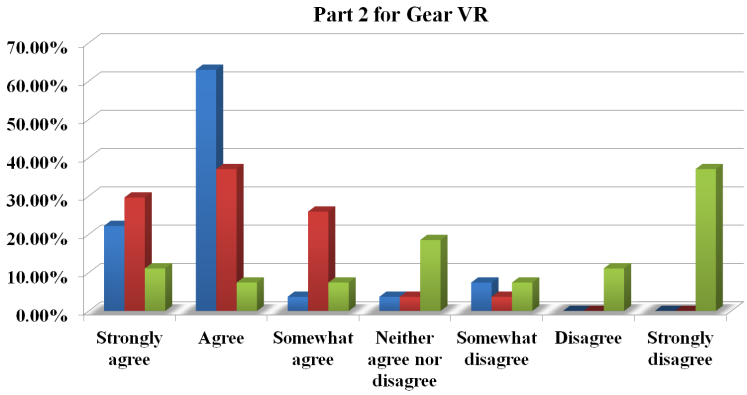
Figure 4.7: Histogram of scores for the Bubble Burst game in case of a) 3D TV, b) Oculus Rift, and c) Gear VR.



(a)

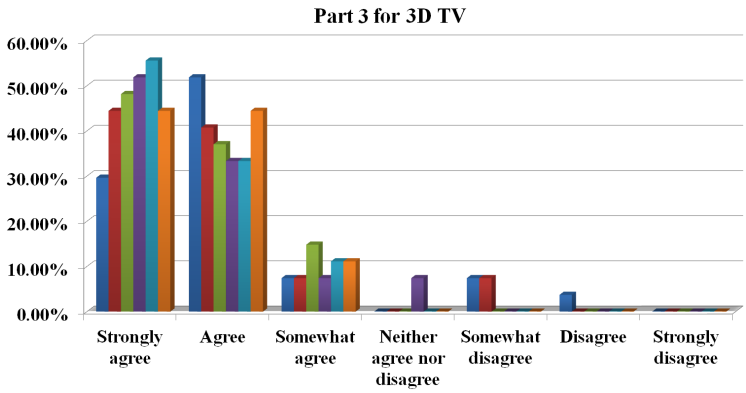


(b)

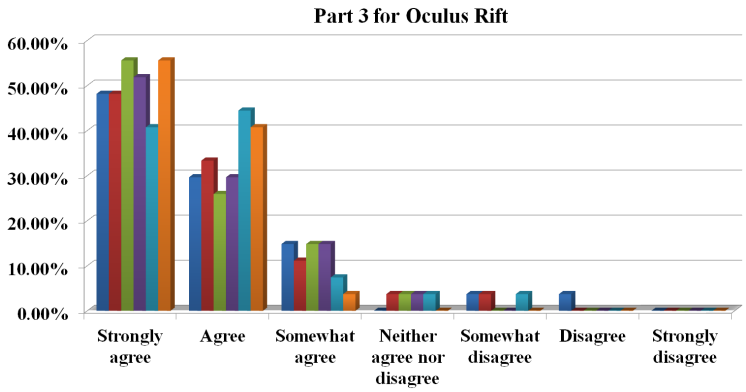


(c)

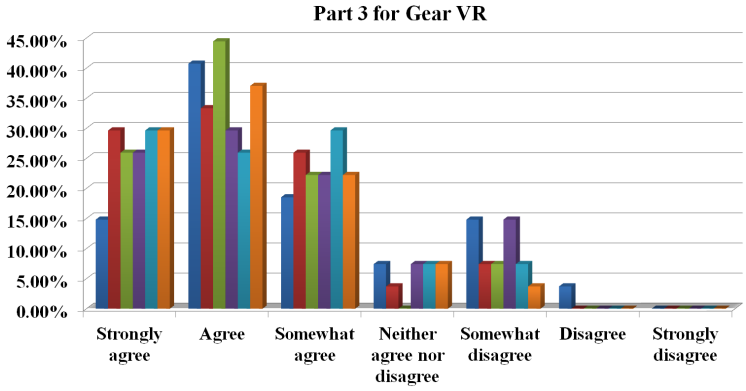
Figure 4.8: Histogram of scores for the phantom movement in case of a) 3D TV, b) Oculus Rift, and c) Gear VR.



(a)



(b)



(c)

Figure 4.9: Histogram of scores for the overall experience in case of a) 3D TV, b) Oculus Rift, and c) Gear VR.

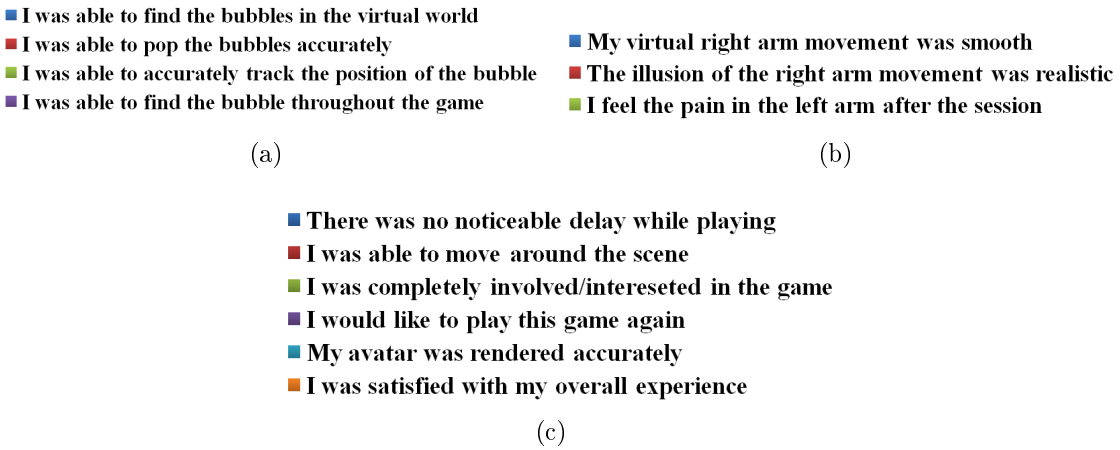


Figure 4.10: Questions and color coding scheme for a) Figure 4.7, b) Figure 4.8, and c) Figure 4.9.

to get familiarized with the virtual game environment. The user study is performed only with the phantom, right upper limb illusion. Initially, to simulate the amputee, we asked the participant to fold their arm and tied it with a bungee cord. As tying the arm with a bungee cord is uncomfortable, we asked the participant to keep their respective hand behind their body to simulate the situation of an amputee. For e.g., if the right arm is selected as a phantom limb, we asked the participant to keep their right hand behind their body. The user study is carried out in three sessions. In the first session, the participant is asked to play the game *Bubble Burst* where he/she can see their mirrored 3D model on the Samsung 3D TV. In the second and third session, the same game is played but with Oculus Rift and Gear VR as a visualization tool respectively.

After each session, they were asked to fill out a questionnaire consisting of total 13 questions that are described in Figure 4.10. The questionnaire was designed to mainly evaluate various factors in the game, overall visual quality, immersive experience, effectiveness of the illusion, realism of the movement and any experience of the latency. As the questionnaire consists of all objective questions, it provides an overall quantitative analysis. The set of questions is broadly divided into three parts: Bubble Burst Game, Phantom movement,

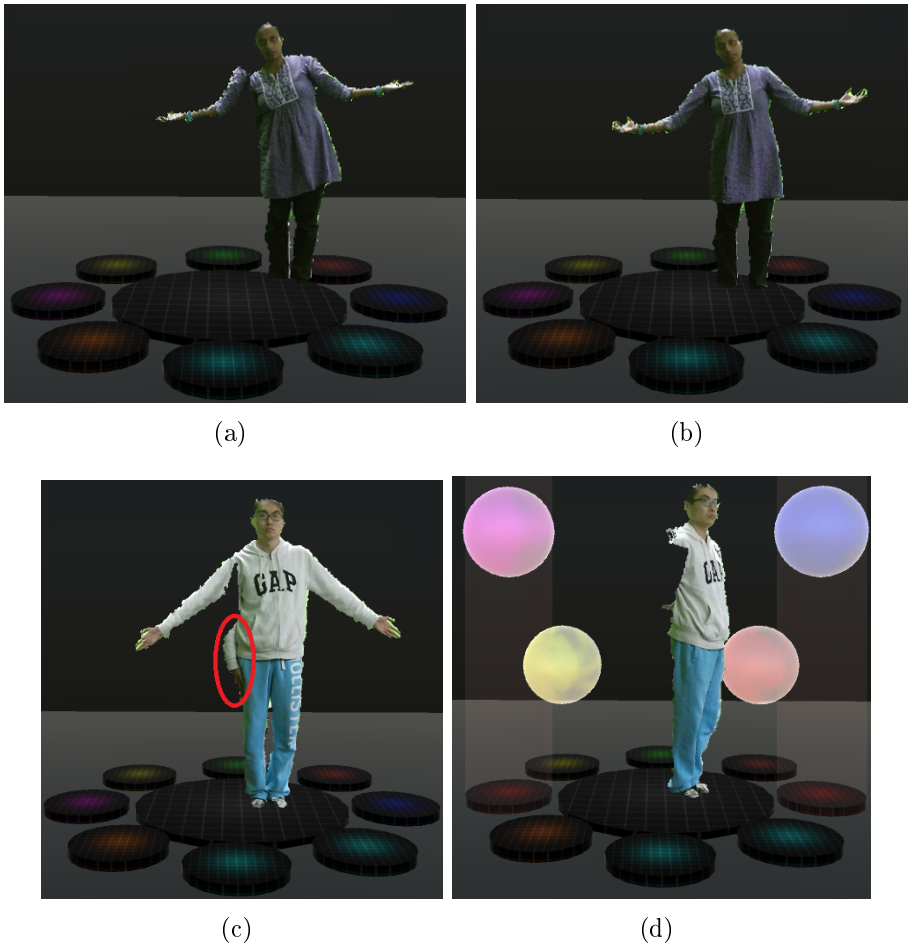


Figure 4.11: a) Using 2D Mirroring, b) using 3D mirroring, c) segmentation Error, and d) effect of occlusion.

and the overall experience. The range of answers is defined by a 7-point Likert-type scale, ranging from 1 being strongly agree to 7 being strongly disagree. The participants were also asked to provide a subjective criticism, both positive and negative to obtain a qualitative analysis. Figure 4.7, 4.8 and 4.9 illustrate the histogram of the scores in different cases. It can be seen from Figure 4.7 that, the majority participants felt the *Buuble Burst* game was easy to play, and they could locate and burst the bubbles accurately. Due to rich, immersive experience in Oculus Rift, a great portion of the participants felt that finding the bubbles were easy with Oculus Rift (100%) compare to 3D TV (92.59%). As illustrated in the Figure

4.8, all the participant agreed that the illusion of the phantom right-hand movement was realistic in the case of 3D TV and Oculus Rift, whereas 92.6% of the participant agreed in the case of Gear VR. As the Mr.MAPP framework needs a participant to move their intact limb throughout the game, it is important to test the stress on the intact limb. We found that the majority participants did not report any pain in their left hand after all the sessions. Figure 4.9 illustrates the rating for the overall experience of Mr.MAPP framework. Most of the participant felt that the overall experience was effective, interesting and engaging. 88.89%, 92.59 % and 74.07% of the participant felt that there was no noticeable delay in case of 3D TV, Oculus Rift and Gear VR respectively. This observation verifies the fact that the limited processing power in Gear VR introduces a significant system latency. Some of the participants also noted that in the case of head-mounted displays one needs to wear the device and hence, it is invasive to some extent. On the other hand, 3DTV provides a complete non-invasive phantom limb illusion.

Subject-Matter Expert Evaluation: To evaluate the usability and effectiveness of the system, we asked 11 Subject-Matter Experts (SME) to participate in the user study and to provide their analysis based on their experience with the real patient with limb amputees. SMEs reported that the Mr.MAPP framework would surely provide a unique experience to the patient with limb amputees. They felt the advantage of the Mr.MAPP framework over the traditional mirror box therapy is the unconstrained smooth movement and the realistic illusion of the phantom limb. Some of the SMEs suggested that the bubble generation in the Bubble Burst game can be modified in the case of lower limb amputee keeping in mind the natural instability during leg movement. Given that the phantom limb pain is closely related to neuroplastic changes, it is difficult to infer the result from this user study with non-disabled participants and map it to a patient with limb amputees. However, SMEs think that innovative and interdisciplinary Mr.MAPP framework with customizable *Bubble Burst* game is suitable for further analysis with limb amputee patients. With the encouraging

feedback from SMEs, we are currently in the process of writing experimental protocols for institutional review board (IRB) approval. After IRB approval, we also plan to study the efficacy of Mr.MAPP framework for patients with both lower and upper limb amputees.

4.6 Discussion

In this section, we discuss various observations made during the course of experiments and based on user study feedbacks.

Correctness of phantom limb movement: As the Mr.MAPP framework performs the mirroring using the local 3D coordinate system of the skeleton, a correct phantom limb representation can be generated. On the other hand, if mirroring is simply performed on 2D depth image, the phantom limb may not be generated accurately when the person has rotated or moved. Figure 4.11 a and b show the phantom limb generated using 2D mirroring and 3D mirroring respectively, when the person has turned sideways.

Dependency on the Skeleton: The procedure of phantom limb generation in Mr.MAPP framework highly relies on the skeleton estimation. Along with the skeletal-based segmentation of the point cloud data, the mirroring of the point is also dependent on the medial axis computed using the skeletal information. Hence, any error in the estimation of the skeletal joints results in artifacts in the resultant phantom limb generation as shown in Figure 4.11 c. Real-time pose estimation methods such as (Shotton et al., 2013; Siddiqui and Medioni, 2010) may fail and provide inaccurate skeleton. On the other hand, accurate skeleton estimation methods such as (Shuai et al., 2017) do not run in real-time. Hence, it is important to design a real-time, accurate skeleton estimation methods that provides a direction for the future work.

Occlusion: If due to occlusion, the intact limb is not visible to the depth sensor, the phantom limb can not be generated because of the lack of data for mirroring. Figure 4.11 d shows the effect of occlusion in phantom limb generation.

First Person View: Two participants with the previous experience of the virtual reality based games commented that the first person view would be beneficial. However, as the 3D data and skeleton of the person are captured from the front side, during the first-person view of the game, when a person moves the hand, the part of the hand may not be visible to the sensor, and consequently, no data is rendered for the occluded part of the arm. As this may result in the degraded experience of the phantom limb, we utilized the third-person view for the *Bubble Burst* game.

4.7 Extending Mr.MAPP for Lower Limb Amputation

Various studies (Demet et al., 2003; Ziegler-Graham et al., 2008) estimating the statistics of different types limb amputations indicate that the majority percentage of the patients are affected with the lower limb amputation. For example, the study described in (Ziegler-Graham et al., 2008) shows that a total of 62% of persons experience an amputation with lower extremity. Subject-matter Expert study performed in (Bahirat et al., 2017) also indicates the large population of the amputee patients in VA suffers from phantom pain due to lower limb amputation. This motivates to extend the existing Mr.MAPP framework to create the illusion of lower limb phantom movement.

Although the majority portion of the Mr.MAPP framework can be directly used, a minor modifications are needed to create a realistic illusion of the lower phantom limb. Firstly, the local skeletal coordinate system should be defined appropriately. Because with the skeletal coordinate system defined at *spine joint* P_{sp}, the shoulder movement may result in the unnatural lower phantom limb generation. Hence, we define a skeletal coordinate system of participant's skeleton with the origin defined at the *hip center joint* P_{hc} (see Figure 4.12) as:

$$x_s = \frac{P_{lh} - P_{rh}}{|P_{lh} - P_{rh}|}, y_s = \frac{P_{sp} - P_{hc}}{|P_{sc} - P_{hc}|}, z_s = \frac{x_s \times y_s}{|x_s \times y_s|}, O_s = P_{hc} \tag{4.8}$$

where P_{lh} and P_{rh} are left and right hip joint, P_{sp} is spine joint, P_{hc} is a hip center joint.

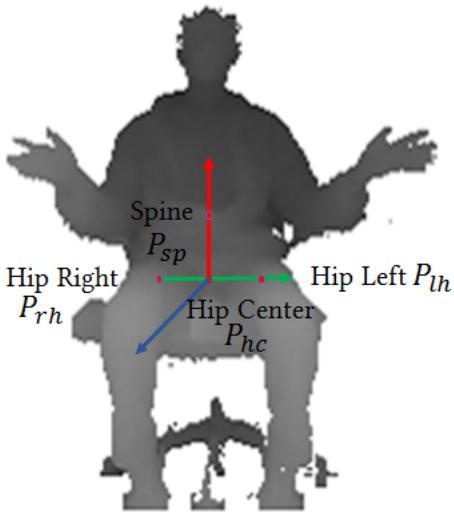


Figure 4.12: The skeletal coordinate system for the lower phantom limb generation.

4.7.1 Issues

However, extending Mr.MAPP framework for the lower phantom limb generation is constrained with several issues such as:

- Jittery skeleton
- Unsymmetrical segmentation
- Small movements remain undetected
- Varying placement in the virtual environment for patients with different physique
- Other person may get detected interrupting the illusion

Jittery skeleton: Although, the Microsoft Kinect provides a reasonable skeleton estimate for a person facing the camera and standing upright, the lower part of the detected skeleton is unstable and jittery for the person in a sitting posture. However, patients with lower limb amputation must participate in virtual games maintaining the sitting posture. Hence, unstable and jittery skeleton becomes a critical issue while extending Mr.MAPP framework

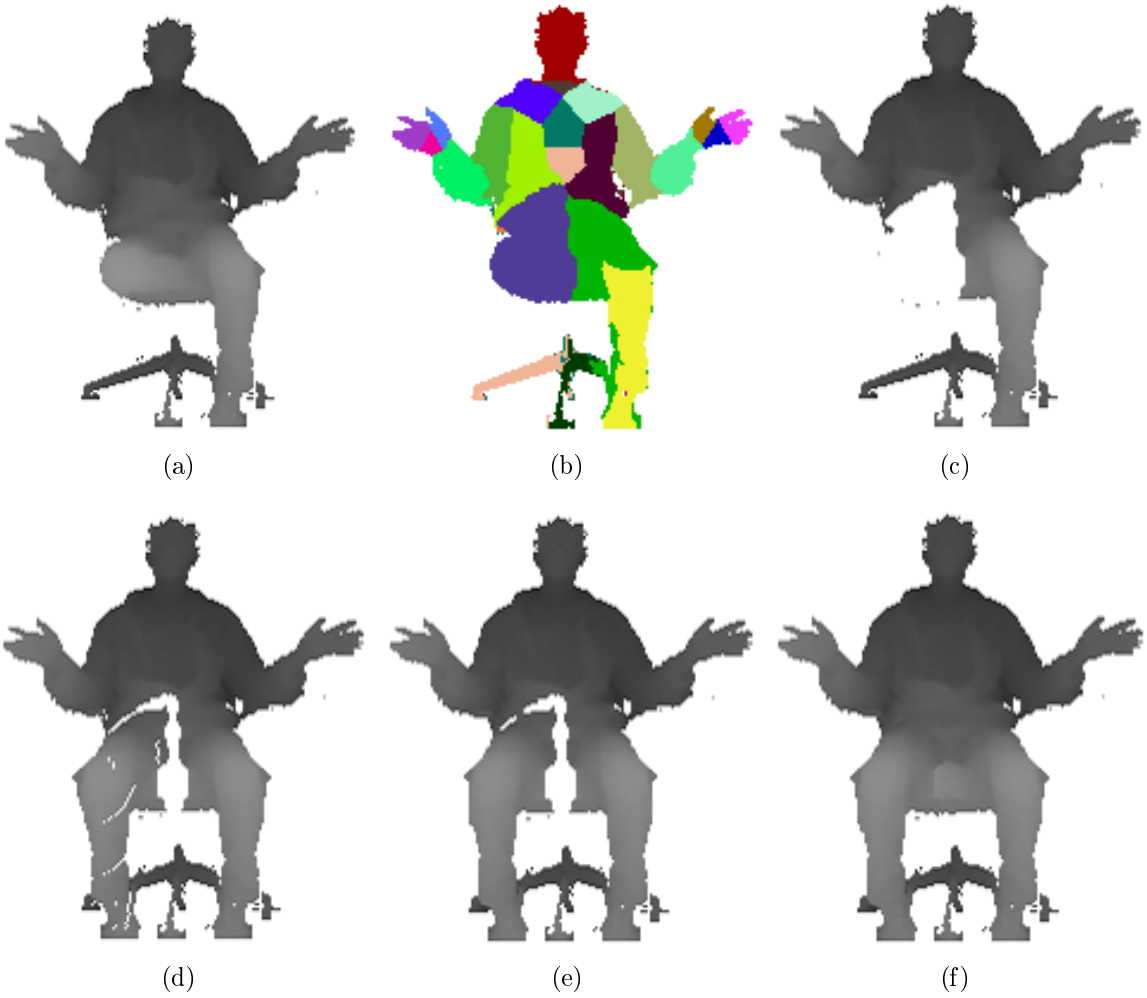


Figure 4.13: a) Original depth image, b) segmented Point cloud, c) modified depth image after removing points corresponding to the right leg, d) modified depth image after adding points belonging to the phantom limb obtain by mirroring the left leg, e)septh image after applying the hole filling and f) result after applying the second hole filling.

for lower limb amputation. To handle the above mentioned instability, we smoothed the detected skeletal joints using an approach suggested in (Azimi, 2012).

Unsymmetrical segmentation: Due to inaccurate skeleton estimation in a sitting posture, the anatomically symmetric joints are detected as unsymmetrical. This unsymmetrical joint estimation leads to unsymmetrical right and left lower limb region as shown in Figure 4.13 b. Therefore, when a part of the affected limb is removed and replaced with the

mirrored intact limb, there may be gaps created due to misaligned affected and intact limb region (See Figure 4.13 d). Such misalignment causes undesirable visual degradation mainly near the root of a limb skeleton hierarchy such as left or right hip joint. Hence, we identify the root of affected limb and fill the holes in the region of interest surrounding it with a symmetrical region from the root of intact limb. Texture coordinates are also appropriately transferred for correct visual rendering. This additional level of hole filling aids in removing the undesired gaps as shown in Figure 4.13 f.

Small movements remain undetected: It is well known that Microsoft Kinect can not detect small movements of end joints such as small palm or foot movement. One of the game designed for patient study (described in Section 4.8.2) requires patient to perform "ankle dorsiflexion" movement. However, as Kinect can not detect such a small movement, the corresponding virtual interaction can be realized. To overcome this issue, we asked patients to wear a "blue color" shoe covers which are readily available. We identify all the points with blue color and mark their centroid as a foot joint. Tracking the blue shoes cover points helps us to track even a small movement of foot joint. Please note that, the care should be taken to avoid having objects of same color as shoe cover in near by vicinity.

Varying placement in the virtual environment for patients with different physique: Unlike, patients with upper limb amputation, patients with lower limb amputation have restricted mobility. Hence, their live 3D avatar must be placed appropriately in the virtual environment for smooth interactions with the virtual objects. If the placement is off by even minor distance, the patients may not be able to interact with virtual objects leading to undesired user experience. However, a single placement strategy will not work for patients with different physique. Hence, we determine the placement of patient's live 3D avatar based on their approximate knee height estimated on the go. We identify the ankle and knee joint of patient's intact limb and determine their reach based on estimated knee height. Based on this estimated reach, the patient is placed in virtual environment

such that the virtual objects are symmetric and at same distance from the hip center joint of the patient and are also reachable with patients foot. Create figure illustrating real-time placement strategy

Other person may get detected interrupting the illusion: Due to limited mobility, the patient with lower limb amputation may need assistance in carrying out the study or in other tasks. Under such circumstances, it is highly likely that other person may enter in the scene while patient is participating in the study. As Kinect randomly filters out a person, the other person may suddenly get detected. This can severely interrupt the phantom limb illusion. Accounting for such possibilities, we identify the person at the center of the scene, mark him/her as a "key person" and track that person based on their body index provided by Kinect. So irrespective of any number of people in the scene, always the "key person" is detected.

4.8 Patient Study

To evaluate the effectiveness and impact of Mr.MAPP framework for managing phantom pain, it is important to test the system with real patients. Hence, we have setup a patient study in collaboration with Veterans Affairs Hospital in Dallas, Texas. We are planning to carry out the patient study involving atleast 10 patients with lower limb amputation.

4.8.1 Requirements

To carry out the patient study successfully and effectively, the system utilized must satisfy following requirements:

Support for lower phantom limb: As larger number of patients suffer with the lower limb amputation, we designed the patient study focusing mainly on managing lower phantom limb pain. Mainly, Mr.MAPP framework is extended to support lower phantom limb as described in Section 4.7. We also have developed three different games involving

movement with the lower limb for making the patient study engaging and fun to participate in.

User-friendly interface: To avoid any impact of stress in system handling on the patient study results, the interface should be easy and user-friendly to operate. We design a simple and intuitive user-interface for navigating through and completing each game session. User is provided with the option of starting the game once the user thinks he/she is ready. There is also an option of exiting the game once the game is over that allows the user to analyze their scores for the current session before exiting the game.

Track the performance: To track patients performance while participating in virtual games, we provide an interface with the database that acts as a "digital diary". With this digital diary, we can track different types of information through out the complete patient study such as: number of sessions completed per day, number of days the user participated in study, his/her performance during each session, session duration and anomalies occurring during sessions. This allows us to check if patient has completed all the intended sessions for complete duration. If patient has any difficulty in playing games and not able to score for pre-define duration then it is considered as "anomaly". We record the duration of anomaly and also video capturing the patient during anomaly that helps us to understand why patient faced difficulties in scoring.

4.8.2 Extended set of games

We design three different games focusing mainly on lower limb movement. Each game is tailored for a specific movement such as:

- Knee flexion and extension
- Ankle dorsiflexion
- Tandem coordinated movement

Knee Flexion and Extension

Posture and movement: In this game, the patient will be sitting on a non-moving chair with a straight posture. The patient needs to perform knee flexion and extension movement as shown in Figure 4.14 a.

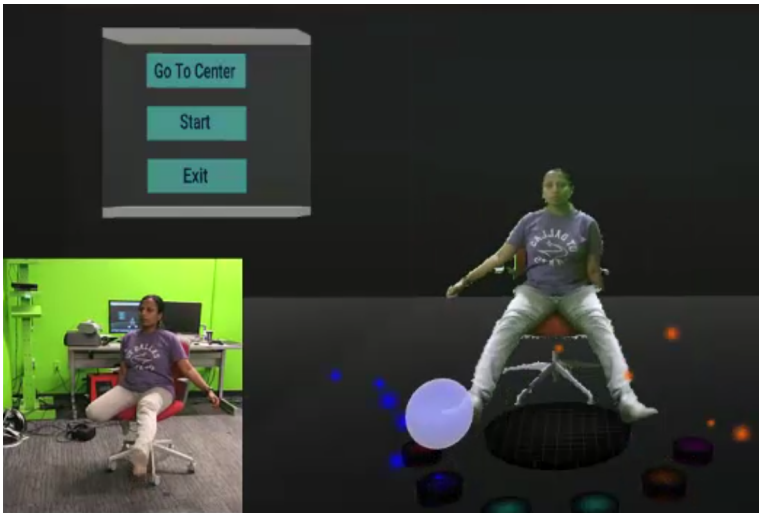
Game setup: This is a modified version of the "Bubble Burst" game described earlier in Section 4.4. In this version, bubble generators on the floor are moved closer to the user so that they are reachable by foot. Figure 4.14 b shows the game layout. As only front two channel are easily reachable with foot, we enable bubble generation only from these two front channels. The goal here is to burst the bubbles while performing the knee flexion and extension in order to save the roof from getting stained with paint. The score will be counted in terms of number of bubbles burst and number of bubbles missed. If the number of bubbles missed is higher that a pre-defined threshold or the time limit is exceeded, then the game is terminated displaying the score of the current session.

Ankle Dorsiflexion

Posture and movement: The second game is called "Pedal" game that is designed to perform ankle dorsiflexion exercise as shown in Figure 4.15 a where patient will be performing a pedaling action. The patient will also required to maintain straight sitting posture. As the pedaling movement is very small, the patient is asked to wear blue color shoe cover (see Figure 4.15 b) and keep their leg on black box or step stool to elevate the foot making it visible to the depth camera.

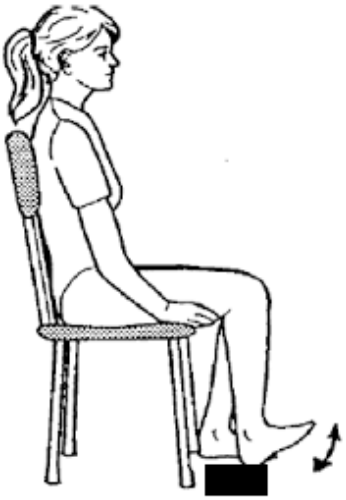
Game setup: In this game (see Figure 4.15), there are pedals in front of patients. As patient presses and releases the pedals, the balls on both sides will start moving upward. The patient needs to keep pressing and releasing the pedals until the balls reach to the top disks. Once the balls hit the disks, it will create a sparkling effect indicating the game is over.

(a)



(b)

Figure 4.14: a) Knee flexion and extension exercise, b) layout of "Bubble Burst" game for lower limb.



Picture Courtesy: https://orthonc.com/uploads/pdf/Ankle_Foot_AROM.pdf

(a)

(b)

Figure 4.15: a) Ankle Dorsiflexion exercise, b) layout of "Pedal" game for lower limb.

(a)

Figure 4.16: Layout of "Piano" game for lower limb.

Tandem Coordinated Movement

In the third game (referred as "Piano" game), the patient is asked to perform a random stomping movement while seated in the chair. There are different piano keys popping up randomly on the floor and patient has to hit the popped key with stomping action. As the correct key is hit, it will create a certain musical note and the key will disappear with sparkles. The score will increase with every correctly hit key. If any key is missed, the score remains unaltered. Once the time limit is exceeded, it is indicated by displaying the "Time Over!" text. Figure 4.16 illustrates the user playing the "Piano" game.

4.8.3 Patient Study Design

Target population: We intend to carry out a study with atleast 10 patients with lower limb amputation. The pool of participant will include diverse set of patients: male and female, different age group, patients with left or right limb amputation.

Study Duration and Session Details: The patient will be participating in the study for 4 weeks. During these 4 weeks, patient will play two sessions everyday where in each session he/she will play three different games. Two sessions are well separated in time, for e.g., first session can be performed in the morning while second can be carried out in the evening.

System Requirements: We use one Microsoft Kinect to capture person's live 3D avatar along with Oculus Rift for immersive experience. The system is setup on a laptop with sufficient processing power. Oculus Rift is also accompanied with corresponding sensors and controllers.

Study Setup: Each patient is first shown the demo of the system during their visit at Veterans Affairs (VA) Hospital, Dallax, Tx. They will be explained all the details about using the system. They will be asked to participate in each game and analyze if they are able to carry out all the tasks. After this session, if they are comfortable using the system and willing to participate in the study, we will ask them to sign a consent form indicating their willingness. If patient agrees to participate in the study, they will be asked to fill out McGill pain questionnaire (Melzack, 1975) to note down their current status of phantom pain. Next, the system will be setup at their home, so that they can perform the intended number of session everyday at their convenience. After completing one week, the patient visits VA Hospital where they will be again asked to evaluate their pain by filling out McGill pain questionnaire (Melzack, 1975). Their digital diary of last one week will be analyzed to find out if they had any difficulty using the system. They will be provided additional guidance in using the system if needed. After completion of four weeks, the complete system is recovered from patient's home and the patient is again asked to evaluate their pain and overall experience during the study.

Current Status of Patient Study: The patient study is currently in progress. As of now, there are two patients participating in the study. We anticipate to complete the

study in a span of one year. The elongated study duration is required mainly because of the following reasons:

- Each patient needs to participate in the study for a duration of 4 weeks. Based on the progress, the patient may be asked to participate in a possible follow up after that.
- Recruiting patients for the study is challenging; mainly due to the lack of awareness about technology and willingness.
- Patients may drop out during the study or can not participate due to other health conditions.
- Patients may have other commitments or personal tasks that may not allow them to participate continuously in the study.

PART III

AUTHENTICATION FOR SECURE USAGE OF 3D DATA

Along with various virtual and mixed reality applications, 3D content is also widely used in pivotal areas such as crime scene reconstruction, self driving cars and 3D surveillance. However, previous research task demonstrates that the 3D content can be easily manipulated. Although, these manipulations are beneficial for multi-platform rendering and virtual therapies, they exposes vulnerability of 3D content. In backdrop of above mentioned sensitive applications and vulnerability of 3D content, it becomes crucial to analyze the 3D content before using it in decisive applications. Hence, in this work we explore the 3D content authentication for secure usage of it.

Chapter 5: To evaluate the efficacy of short range RGB-D cameras, we propose a novel framework that attacks the live 3D depth data stream to manipulate the 3D surveillance feed. We also develop a depth noise analysis based method to identify forgery attacks.

Chapter 6: For long range LiDAR sensors, we carried out a study that explores various attacks on LiDAR data. We also developed two novel forensic algorithms to detect additive attacks on LiDAR data based on inconsistency in point density and inconsistency in occlusion effect.

Chapter 7: Focusing mainly on self-driving car application, we designed a ALERT (Authentication, Localization and Estimation of Risks and Threats) that adds a secure layer in the decision support used in Advanced Driver Assistance System (ADAS). It provides efficient and effective dynamic watermarking scheme for tamper-proofing 3D LiDAR data. The enhanced security layer is facilitated with cross-modal authentication and risk factor assessment.

CHAPTER 5

EVALUATING THE EFFICACY OF RGB-D CAMERAS FOR SURVEILLANCE[1]

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

[1]

Figure 5.1: The behavior manipulated rendering of a person.

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 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 5.1). (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., 2013) 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).

5.1 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

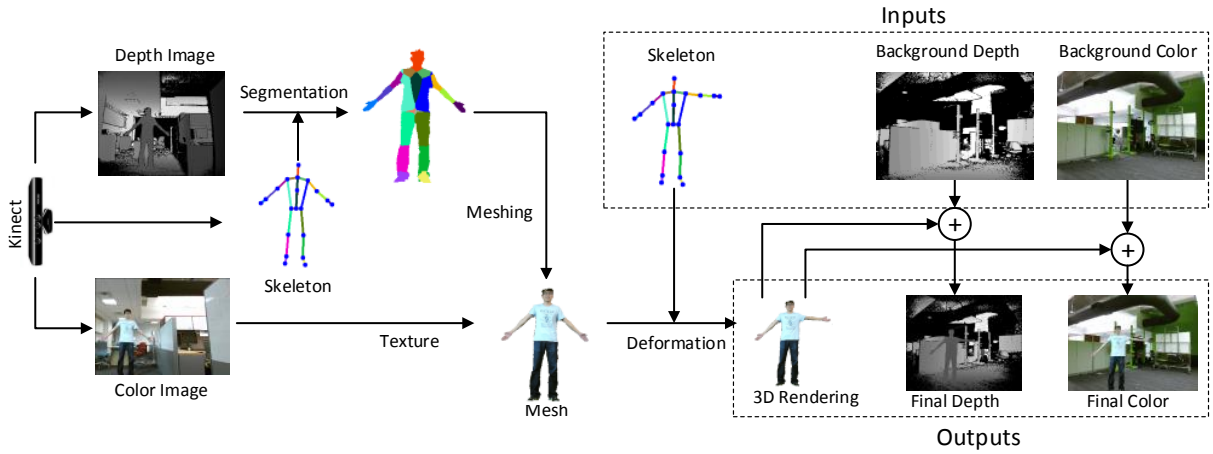


Figure 5.2: The various aspects of anti-forensic framework to generate behavior manipulated streams.

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.

To generate the forged information, color, depth and skeletal streams of the Kinect are used. As shown in Figure 5.2, 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.

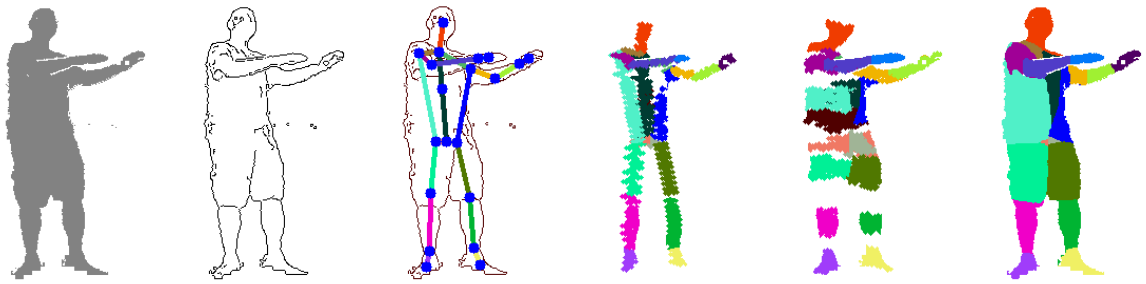


Figure 5.3: The various stages in the segmentation process – (from left to right) the depth image is filtered to extract the person, then edges are estimated, skeleton is overlaid on the edge image, segments are then identified by growing from the seed points on the joint, reaching the final result.

5.1.1 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 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 5.3 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 5.3.

Skeleton identification is performed using a real time, highly accurate depth image based approach (Shotton et al., 2013) 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

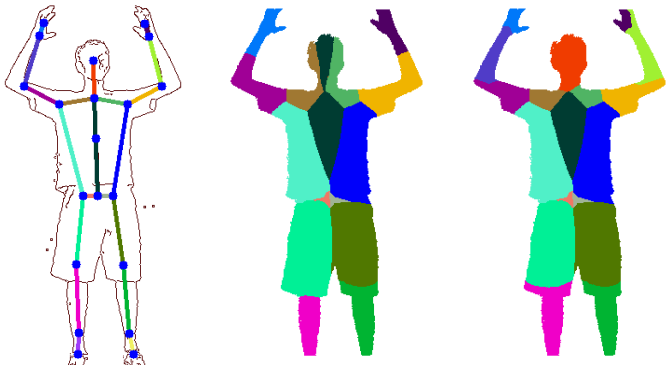


Figure 5.4: The effect of number of seeds points on the segmentation, from left to right the skeleton is misaligned a little, using a single seed point at the middle of the joint results in bad segmentation and using the multiple seeds fix gives optimal segmentation.

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 5.4.

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 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.

5.1.2 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

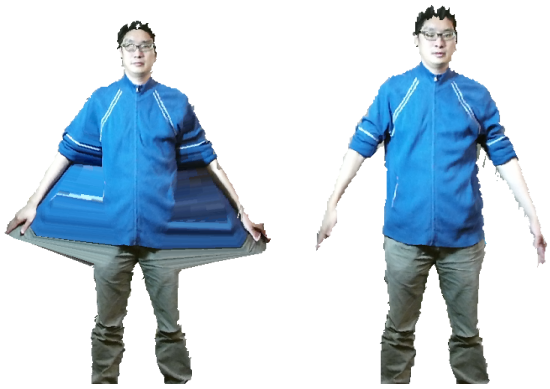


Figure 5.5: Deformed meshes generated without using segmentation information are on the left, and with segmentation information on the right.

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 (Pajarola et al., 2003) are 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 5.5.

Skeletal animation is performed on the segmented model using rigid body deformation. The real time streaming deformation technique described in (Raghuraman et al., 2013b) 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 5.6. 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.

5.2 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

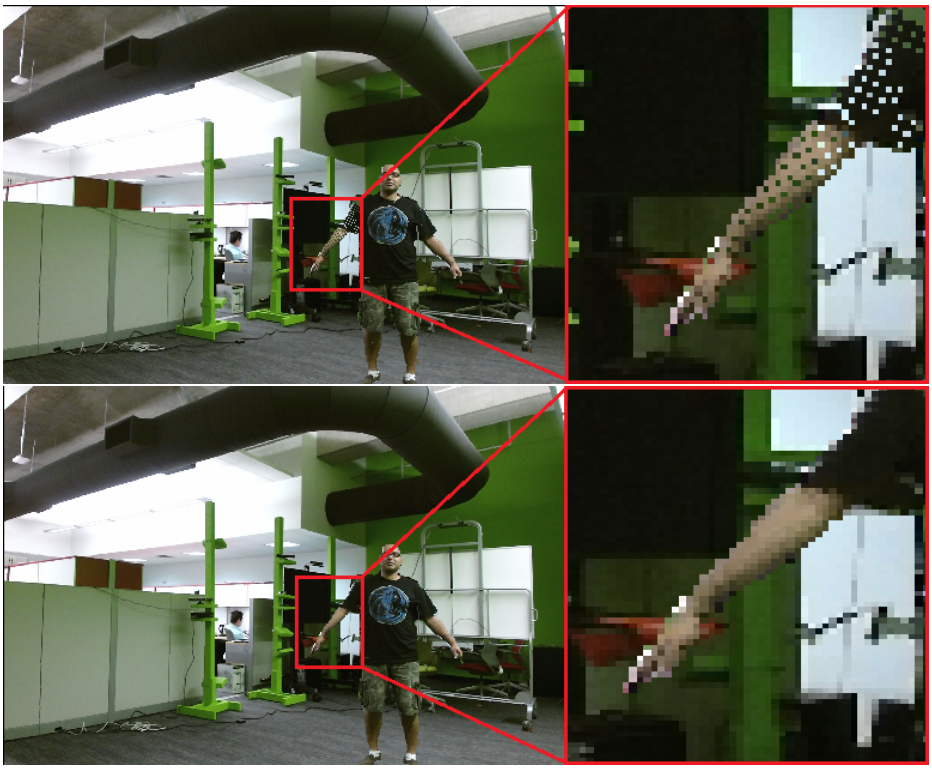


Figure 5.6: Altered color image generated without interpolation on top and with interpolation on bottom.

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 5.4. Using standard distance based techniques, like Voronoi diagrams, for estimating segments as described in (Raghuraman et al., 2013b), result in bad segmentation when the bones are close to each other as shown in Figure 5.7.

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.

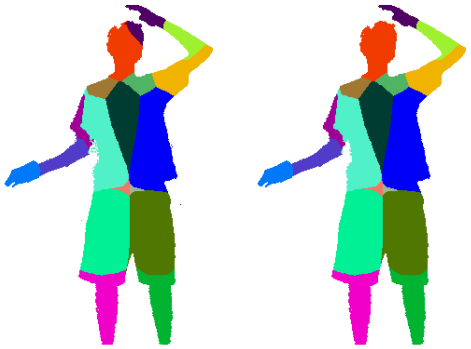


Figure 5.7: The segmentation done using Voronoi on the left and edge based method on the right, notice the head region.

5.2.1 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

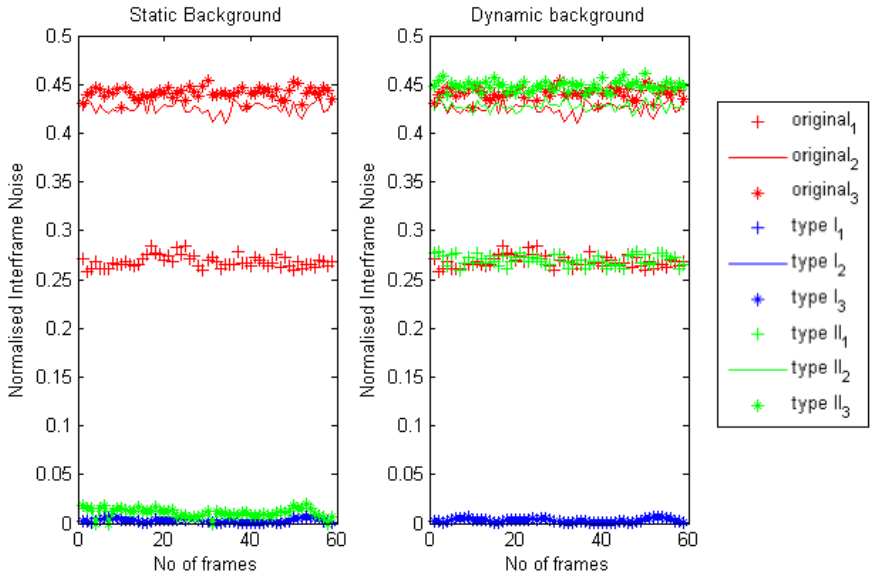


Figure 5.8: Noise analysis for original depth images, Type I and Type II forged depth images with single image and multiple images as background in all frames.

frames. Following plots assist in understanding the estimated noise ranges in all the cases. Figure 5.8 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 5.8, 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.

5.2.2 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.

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.

5.3 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.

CHAPTER 6

A STUDY ON LIDAR DATA FORENSICS [1]

6.1 Introduction

With recent advances in the depth sensing technology, it has become possible to quickly generate a complete 3D reconstruction of an object or an entire scene. Various depth sensors such as LiDAR are widely available in the market which can be used to scan indoor and outdoor scenes. Due to low cost, millimeter precision and ease of operation, the 3D scanned data obtained using the LiDAR sensors finds application in diverse areas (B., 2014; Kashani et al., 2014; Chen et al., 2013). Benedek (B., 2014) demonstrates the capability of rotating multi-beam LiDAR as a future surveillance camera for a real-time 3D people surveillance. Various studies performed (Kashani et al., 2014; Chen et al., 2013) encourage to use LiDAR data for damage detection in case of large deformed structures such as bridges, roofs. 3D laser scanning has become a powerful tool to collect the crime scene and civil accident data and bring it to the courtroom for legal investigation or insurance settlement (W. et al., 2013; Francis, 2006; Jones, 2011). This technology allows for a collection of 3D data of the scene where the civil or criminal incident took place and to create the same scene graphically in a courtroom. It also allows permanent 3D archive of a scene involved in judicial proceedings which can be referred in future as well. Further, LiDAR has been successfully employed in autonomous automated vehicles such as Google Driverless Car (Guizzo, 2011), Stanford Shelley (Levinson et al., 2011), Annieway (Stiller and Ziegler, 2012) for detecting the obstacles and re-planning the mission/path accordingly.

In this context, the genuineness of the 3D LiDAR data is a critical factor which further motivates to determine the possibility of forgery attacks on it. Any attack that manipulates

[1] ©2017 IEEE. Reprinted, with permission, from K. Bahirat and B. Prabhakaran, "A study on lidar data forensics," 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, 2017, pp. 679-684. DOI: 10.1109/ICME.2017.8019395

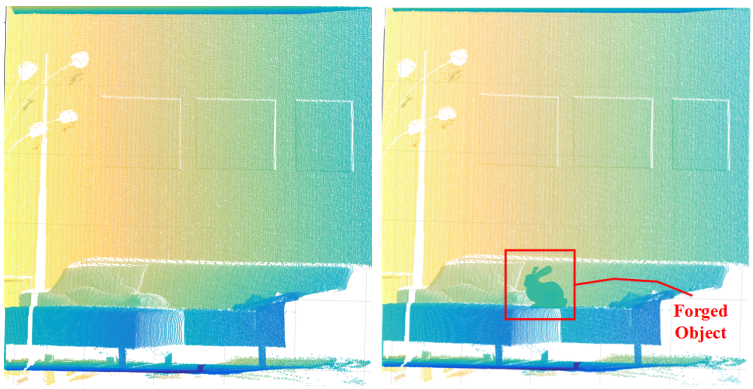


Figure 6.1: Original LiDAR scan (left) and forged LiDAR scan (right).

the LiDAR data can be very harmful in the above applications. For example, in the case of autonomous automated vehicles, a false indication of an obstacle can cause a wrong driving decision and can potentially lead to an accident. Similarly, adding an object such as a fake gun or weapon in crime scene may result in incorrect judgment. Hence, it is important to address following two questions:

- Is the LiDAR data vulnerable to forgery attacks?
- Is it possible to detect such forgery attacks on the LiDAR data if there exists any?

In this paper, we address these questions as follows:

- We identify three possible approaches for attacks on the LiDAR data that do not need additional commodity hardware. Experimental results show the successful *blinding* due to proposed attacks. Figure 6.1[2] shows original and forged LiDAR data.
- We also present two novel algorithms for detecting such forgeries in LiDAR data and provide a detailed performance analysis of the proposed algorithms in case of different types of attacks.

[2]Note, all the images in this paper are better visualized in color.

Principal Contributions: This paper provides a detailed study of possible forgery attacks on LiDAR data. It outlines two novel forensic approaches for LiDAR data (As per our knowledge, this is the first attempt to address LiDAR forensics). Though the forensic algorithms are effective for specific types of forgeries, a forensic approach handling the wide spectrum of forgeries is necessitated. This work creates awareness about avoiding the blind usage of LiDAR data in critical applications and motivates to explore forensic of LiDAR data as an emerging research area.

6.2 Related Work

Forgery detection in images/videos has been a very well researched area. Two excellent surveys (Farid, 2009; Milani et al., 2012) provide a list of current state-of-the-art methods in image/video forensics and highlight their features. On the other hand, forensics for 3D data is a relatively less explored area. As per our knowledge, no forensic method has been proposed to detect forgery in 3D data except the method proposed in (Raghuraman et al., 2015b). Raghuraman et al. (Raghuraman et al., 2015b) propose an anti-forensic framework to capture and manipulate the live RGB-D data stream to create a realistic illusion of an individual performing the activities which they did not actually do. Authors also suggest a preliminary noise analysis based forgery detection for depth images which is incapable of detecting forgeries in all cases and not suitable for LiDAR data.

In literature (Qi, Dong-qing, and Da-fang, Qi et al.; Medimegh et al., 2015), a verity of methods based on digital watermarking are proposed for 3D models. Most of these methods are designed for polygonal meshes that have two categories: *robust* and *fragile methods*. *Robust methods* (Wang et al., 2012; Jing et al., 2014) are constructed with the aim of providing ownership protection and distribution channel tracking. While *fragile methods* (Yeo and Yeung, 1999a; Lin et al., 2005) are designed for authentication applications. As these methods require the connectivity information, they are not suitable for LiDAR data.

Most of the robust watermarking methods for 3D point cloud (Agarwal and Prabhakaran, 2009; Qi, Dong-qing, and Da-fang, Qi et al.) providing copyright protection involve expensive clustering based techniques or need to find ordering in points which make them inapplicable to authenticate LiDAR data quickly.

The resilience of a LiDAR against attacks has been studied with respect to the security analysis of an automotive system (Petit et al., 2015; Petit and Shladover, 2015). Petit et.al. suggest a use of a smart surface which is absorbent or reflective in nature to manipulate the data sensed by the LiDAR in (Petit and Shladover, 2015). Relaying and spoofing attacks on LiDAR sensor with the aim of generating fake echoes and fake objects have been proposed in (Petit et al., 2015). In "relaying" attack, the original signal sent from the LiDAR is relayed from the other position to create fake echoes with additional two transceivers. A "spoofing" attack is made by sending a counterfeit pulse during a listening interval of 1.44 microseconds of LiDAR to create an illusion of point being further away. Most of the work aim at studying possible attacks on the LiDAR sensor and hence require additional hardware.

We differentiate from the work mentioned above by developing *attacks on the 3D LiDAR data* that do not require additional commodity hardware.

6.3 Background on LiDAR data

To analyze the possible threats to LiDAR data, we need to understand the structure of the LiDAR data and the nature of applications employing it.

LiDAR Data: LiDAR is a depth-sensing technology that measures distance by projecting a laser beam on the target. Generally, LiDAR scanners use the *time of flight* approach or *phase difference* approach for computing the distance. These scanners typically have a larger range, and the accuracy of the system varies between 3 to 8mm (W. et al., 2013). Typically the data obtained using LiDAR scanner is stored in the form of the list of points repre-

sented using their X, Y and Z coordinates (also termed as "unstructured 3D point cloud"). Advanced sensors also provide a color information associated with each point.

Classification of LiDAR applications: Applications using LiDAR data can be broadly classified into two categories.

- *Visualization Applications* consist of applications where data is directly consumed by a human user through visualization; for example, crime scene reconstruction, damage detection. In such applications, the LiDAR data is processed offline to generate a surface reconstruction from the 3D point cloud data for visualization purpose.

- *Automated Decision Systems* include applications such as automotive vehicles where the LiDAR data is processed by an algorithm instead of a human user. In this case, the 3D point cloud data is used by an algorithm to determine a path or to make a decision for navigating the vehicle.

Based on the applications, the possible attacks on the LiDAR data have different levels of complexity in terms of time taken and active involvement of attacker which is illustrated later in Section 6.4.5.

6.4 Anti-forensic Framework

To evaluate the vulnerability of the LiDAR data, we propose a novel anti-forensic framework that utilizes basic computer graphic techniques to create forged LiDAR data. Adopting the attacker model used in (Petit et al., 2015), we assume that the attacker has limited resources in terms of the type of LiDAR sensors and processing power and has the intention to disrupt the data unnoticeably. The proposed anti-forensic framework is designed to create three types of attacks on the 3D LiDAR data. Figure 6.2 shows the pipeline for different types of attacks on LiDAR data.

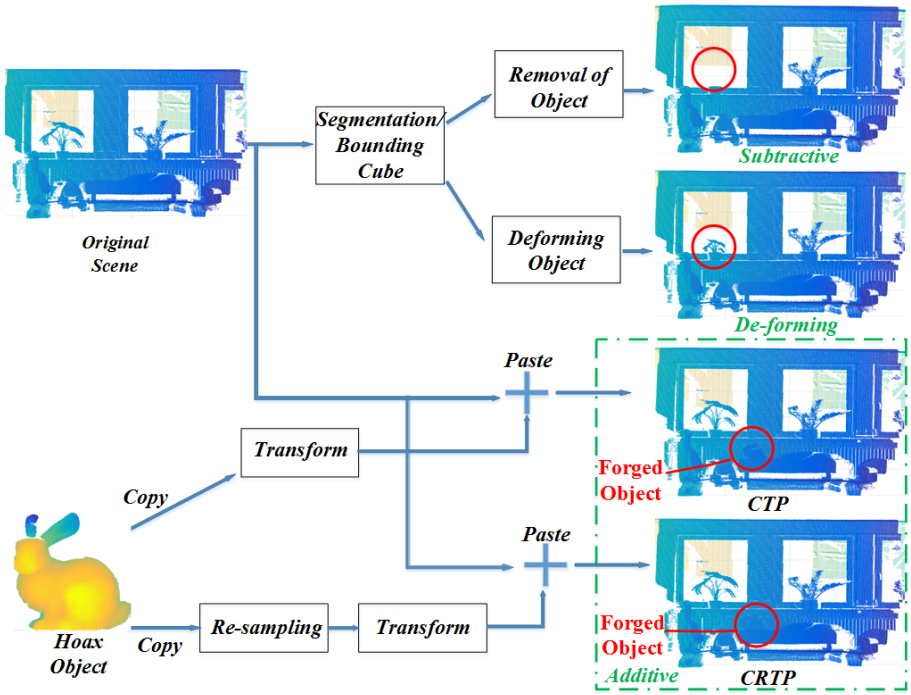


Figure 6.2: Pipeline for different types of attacks on 3D LiDAR data.

6.4.1 Additive attacks

In additive approaches, an object is inserted and placed in the original scene such that the viewer or an automotive system perceives the object being actually present in the scene. Acquiring the inspiration from the copy-paste forgery in digital images (Fridrich et al., 2003), we design two additive approaches for attacking LiDAR data as follows:

Copy-Transform-Paste (CTP)

Similar to the copy-paste forgery attacks on the image, to add a hoax object in a scene, one can copy the set of points corresponding to the object and add it to the list of points corresponding to the scene. But this preliminary approach may not create an impactful illusion due to following factors:

- Due to the limited resource availability to the attacker, we can assume, without the loss of generality, that the attacker may have a different sensor than the one used for scanning the scene. Two different sensors may utilize different metric for measurement. Hence, a range of coordinate values of hoax object may be significantly different than that of the scene.

- 3D coordinates of the points representing the object are defined with respect to it's local coordinate system which may be different from that of sensor used for scanning the scene.

- The orientation of the local coordinate system of the object may differ from its orientation in target scene based on its placement.

To handle the issues as mentioned earlier, we incorporate an additional step of *"Transform"* before pasting the points of a forged object into the scene. It consists of following transformations:

Scaling: Scaling helps to resolve the issue occurring due to a difference in metric used by different sensors for measurement. The scaling factor can be computed based on the knowledge of the metric employed by each sensor or based on the range of values for the coordinate of the set of points representing the scene and object.

Translation: The points representing the object are first required to be translated into the local coordinate system of the scene followed by a translation needed to place it in the correct position in the scene. The cumulative amount of required translation t can be computed as $t = P_{object} - (o_{scene} - o_{object})$, where o_{scene} and o_{object} are centroids of the scene and the object respectively and P_{object} is the position of the object in the scene.

Rotation: The object needs to be rotated as per the required orientation of the object in the scene. It can be achieved by multiplying each point of the object with the rotation matrix. Note that, rotation required is decided by the attacker based on how the object is supposed to be placed.

Copy-Re-sample-Transform-Paste (CRTP)

The second attack is designed based on the fact that two depth sensors may have different resolution based on underneath hardware. In Copy-Re-sample-Transform-Paste forgery, we add one more step of *"Re-sampling"* before applying *"Transform"* step. In this step, the forged object is *"re-sampled"* i.e., it is either downsampled or upsampled to match the resolution of the sensor used to scan the scene. Downsampling can be achieved by employing any of the point cloud sampling methods such as uniform sampling or Poisson disk sampling (Hou et al., 2015). Upsampling can be accomplished by performing interpolation of the current samples. Next, we define the minimum inter-point distance (MID) which is Euclidean distance between the point, and it's nearest neighboring point. The factor of re-sampling is defined as:$\gamma = \frac{MID_o}{MID_s}$, where MID_o and MID_s are MID between points of object and points of scene respectively. This definition of the re-sampling factor allows removing inconsistency occurring in sampling density due to insertion of a new object.

6.4.2 Subtractive Approaches

In subtractive forgery, to conceal the presence of an object in the original scene, the set of points representing the object are removed from the LiDAR data. Due to the unstructured nature of LiDAR data, identifying the points belonging to an object is a nontrivial task. Identification of the objects can be made manually by selecting a bounding cube around it with the help of visualization toolkit or by performing a point cloud segmentation using the method described in (Zheng et al., 2013). Segmentation provides the labeling for each object in the scene. Hence, points having the same label can be eliminated from the original scene to perform subtractive forgeries. On the other hand, given the spread of a bounding cube, an algorithm determines the set of points inside the bounding cube and removes them. It should be noted that the segmentation will significantly increase the complexity of attack in terms of the time and efforts needed.

6.4.3 Deforming Approaches

In deforming approaches, the point representing the portion of the object are displaced from their original position. This type of forgery can be used to create a fake dent on the surface of the object. The identification of the object to be deformed can be performed using either manual selection of bounding cube or the point cloud segmentation. Points can be displaced in a random fashion to achieve the deformation. As deforming attacks mainly target the data used in the visualization based applications and complexity involved in identifying the portion of an object to be deformed make this attack more complicated.

6.4.4 System Overview

The proposed system allows user to select the attack type. If the additive attack is selected, the user needs to provide a 3D model of the hoax object, position and orientation of the object in the scene. Based on the required position and orientation, the parameters needed for *"Transform"* step such as rotation matrix and translation vectors are computed only once. Further, if a user selects to re-sample the data, the factor of re-sampling is obtained as suggested in the Section 6.4.1. To compute the re-sampling factor, the data in the vicinity of the targeted location of the hoax object is obtained using the rotation and translation. For subtractive and deforming attacks, the user needs to provide a bounding box indicating the targeted area and deformation scale. For example, the user can decide to remove any object at a distance of 10 meters in front of the car with length, width, and height of 2, 3 and 5 meters respectively. Using these input parameters the system can generate a selected attack.

6.4.5 Complexity of Attacks

Above mentioned attacks differ in complexity based on the type of application utilizing LiDAR data. For example, additive attack on LiDAR data for the autonomous vehicle will

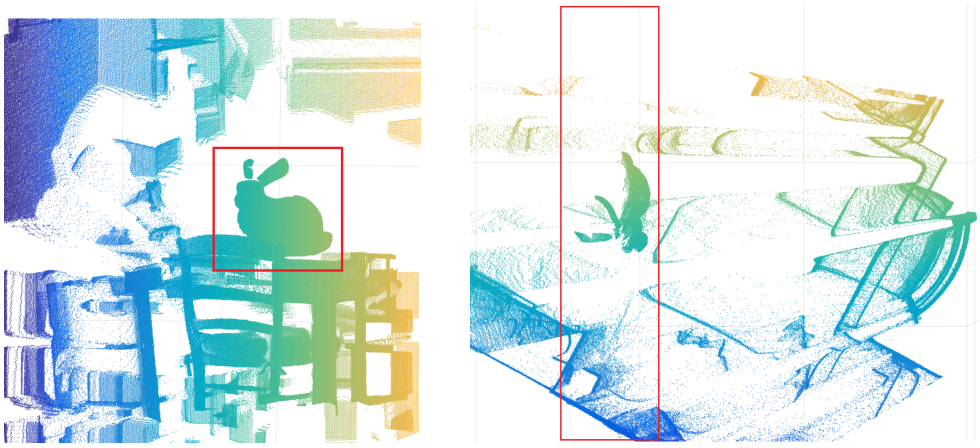


Figure 6.3: Front view (left) and top view (right) of the forged scene illustrating variation in sampling density based on distance from sensor.

be easier to implement compared to the additive attack on the LiDAR data used for crime scene reconstruction. In the case of crime scene reconstruction, extra care must be taken while placing the forged object in the scene as the final reconstruction is visualized by a human user. To create an impactful forgery, orientation, and placement of the object should maintain the law of physics such that scene appears to be natural. On the other hand, in the case of an automotive vehicle, a mere presence of hoax object alters the decision of algorithm and does not require intensive human intervention while performing an attack. In this paper, we mainly consider attacks on the LiDAR data used for applications where data is used by algorithms and not by the human user.

6.5 Forensic Evaluation

The detailed analysis of the possible attacks on LiDAR data motivates to design an algorithm to validate LiDAR data. We now describe two preliminary forensic algorithms for additive attacks. Due to the page limit, we restrict ourselves to forensic evaluation of additive attacks only.

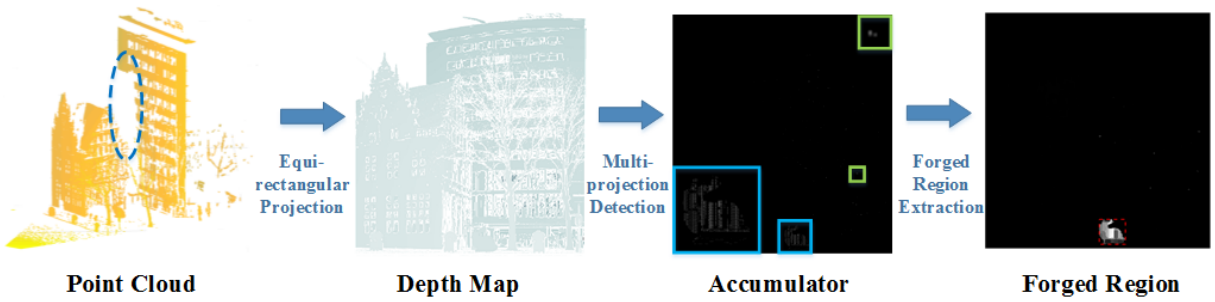


Figure 6.4: Pipeline of *IsOcclusionConsistent* (forensic algorithm II). Input to the algorithm *IsOcclusionConsistent* is the forged point cloud obtained by adding Stanford Bunny scan into the one of the outdoor scene scanned using LiDAR.

6.5.1 Density Variation Based Forensic Algorithm I

LiDAR sensors provide a discretization of an object surface. As resolution varies across sensors, the discrete point clouds of an object obtained using different sensors will have different sampling densities. Further, due to perspective projection based design of depth sensors, the sampling density of an object also depends on its distance from the sensor. For example, objects near the sensor will have a higher sampling density compare to objects away from the sensor. Moreover, the objects at the same distance from the sensors must have approximately similar sampling density.

Assuming that the additive attacke is created using different sensors with different resolution, the forged object will have a different sampling density compare to another object in the scene at the same distance from a sensor as that of the forged object. Hence, if we compute an MID (Minimum Inter-point Distance) at different values of z (value of z represents the distance from the sensor), it will increase as we move away from the sensor if the data is unaltered. On the other hand, if suddenly there is a continual rise or drop in MID for the range of depth values, then we can consider it as forged data. Figure 6.3 illustrates the sudden variation in the MID in the case of forged scene. It can be seen that inter-point distance corresponding to bunny is very high compared to other parts of the scene. Based on these observations, we formulate the algorithm *IsDensityConsistent* that checks the consis-

tency of sampling density. Here, P is the point cloud of original scene, $d(p, p')$ is Euclidean distance between p and p' and z represents Z-coordinate of p.

Algorithm 1 *IsDensityConsistent*

for each point $p \in P$ **do**
    Compute nearest neighbor p'
    Compute $min_d(p) = d(p, p')$
end for
Sort points in the increasing order of z values.
Quantize z values to set of discrete levels Z
for each discrete level $z_1 \in Z$ **do**
    Consider set of point Q at distance z_1
    Compute $MIN_{ipd}(z_1) = mean(min_d(p))$ for all $p \in Q$
end for
Compute Moving average of the signal $MIN_{ipd}(z)$
if sudden rise or the fall in the averaged $MIN_{ipd}(z)$ **then**
    Declare "Forgery Detected"
end if

6.5.2 Multi-projection Based Forensic Algorithm II

Generally, due to the occlusion effect, objects which are behind the object closer to the LiDAR sensor may not be entirely or partially visible to the sensor. It can be observed from Figure 6.4 that the portion of the taller building is occluded by the front building and no points are measured by the LiDAR sensor in the occluded region. On the contrary, when the hoax object is inserted into the scene with additive attacks, the forged data will not exhibit such occlusion effect. Hence, points behind the hoax object still exist causing inconsistency in the occlusion effect characterising the scanned data.

Inspired by this idea, we propose a multi-projection based forensic algorithm, *IsOcclusionConsistent* (also referred as II) that determines the validity of the 3D LiDAR data by checking congruity in the occlusion effect. Figure 6.4 illustrates the pipeline of the proposed forensic algorithm II that includes following steps:

Equi-rectangular Projection: For the unaltered LiDAR data, when all 3D points are projected on the 2D image plane, each point will be mapped to a distinct image pixel based on the selected image resolution. On the other hand, when a hoax object is inserted into the scene, there might be multiple points getting mapped to a single image pixel. We leverage this fact to determine if there is a point behind the current point i.e., to determine if there is any inconsistency in the occlusion effect.

To obtain the projection of the scanned 3D data on a 2D image plane, we first convert each 3D point (x, y, z) in the Cartesian co-ordinate system to the corresponding (r, θ, ϕ) in the spherical co-ordinate system as follows:

$$r = \sqrt{x^2 + y^2 + z^2}; \theta = \arctan(\frac{y}{x}); \phi = \arccos(\frac{z}{r}) \tag{6.1}$$

where r, θ and ϕ are radial, azimuth and zenith coordinates.

Next, we apply a equirectangular projection (Houshiar et al., 2015) that relate the 2D image coordinates (i, j) linearly to θ and ϕ i.e., $i = \theta$ and $j = \phi$. This projection supports 360°in the horizontal and 180°in the vertical field of view. The resolution of the image is determined by the vertical and horizontal angle resolution of the LiDAR sensor. Figure 6.4 shows the depth map generated using the equi-rectangular projection.

Multi-projection Detection: To determine if multiple points are getting mapped to a single image pixel, we maintain a data structure, called as "Mask". If the 2D projection of the current point p is already occupied by another point q, we compute the distance between these point along the projection line as $|r_p - r_q|$ and accumulate it in the "Mask" at their common 2D projection (i, j). The mask obtained using the proposed multi-projection detection is shown in the Figure 6.4. It can be seen that the bunny shape region has higher accumulation density. Therefore, there exist multiple points behind the bunny in the scanned data evidences the forgery.

Forged Region Detection: Though, the "Mask" obtained during the multi-projection detection captures the forgery, it also consists of few artifacts due to a slight mismatch

between the image resolution and resolution of the LiDAR sensor used for scanning as well as the resolution of the forged object. The top, right box in the mask shown in Figure 6.4 describes such noisy points. While, bottom-left box shows the sparse bunny image. If we estimate connected components in the mask, the results may not accurately represent the forged region.

To annihilate above mentioned issues, we apply post-processing to "Mask" that includes morphological closing and median filtering followed by connected component estimation. The presence of a connected component with the area greater than the empirically defined threshold A_{th} is considered as a forgery.

6.6 Experimental Results

In this section, we demonstrate the effectiveness and efficiency of the proposed additive attacks and the proposed forensic evaluation algorithm. All the attacks and algorithm are implemented in MATLAB and experiments are run on a CPU with Intel (R) Core (TM) i7-5820K with 3.30GHz speed and 32GB internal RAM.

We utilize LiDAR scans provided in the Robotic 3D repository (Nüchter and Lingemann, 2010) along with Bunny and Dragon models provided in the Stanford Dataset as hoax objects. Robotic 3D repository consists of both indoor and outdoor scenes which are scanned using Riegl VS-400 and an Optris PI IR camera. For our experiments, we consider indoor scenes which are taken at a residential house in Germany and outdoor scenes which are taken at downtown Bremen. From this dataset, we randomly select 44 scenes which consist of 27 outdoor scenes and 17 indoor scenes. We create 42 CTP type forgeries by adding Bunny and Dragon models in 13 different outdoor scenes and 8 indoor scenes each. Next, we create 24 CRTP types of forgeries by inserting re-sampled Bunny and Dragon models in 7 outdoor scenes and 5 indoor scenes. We also create 10 subtractive and deforming forgeries by identifying the object using MATLAB visualization tool. Figures 6.5 and 6.7 show the

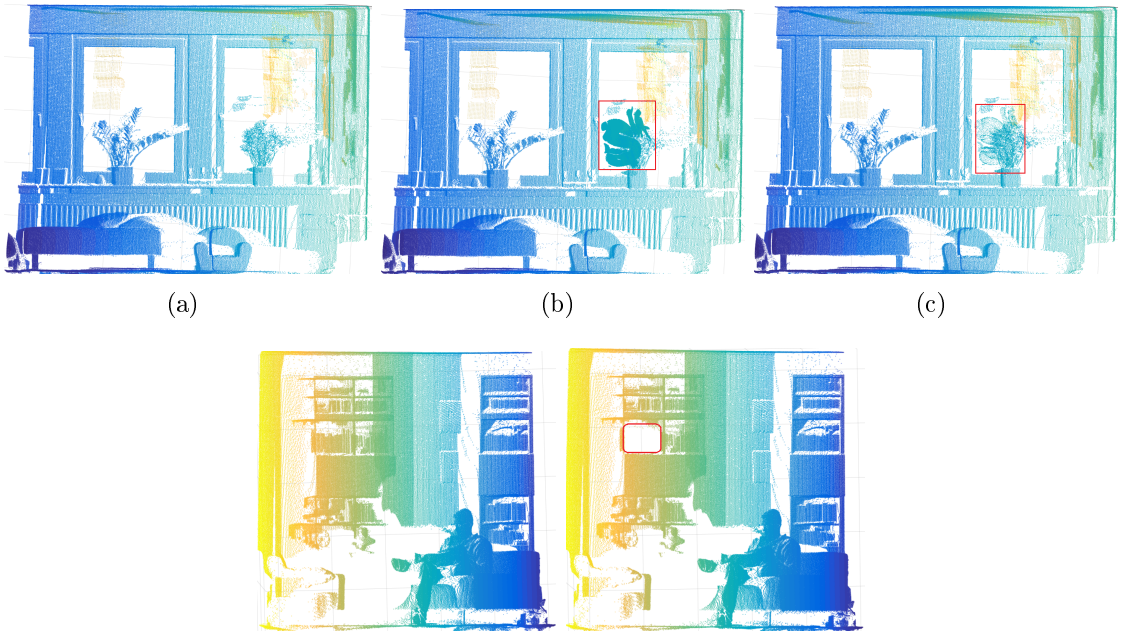
visualizations of CTP, CRTP type forgeries and subtractive, deforming forgeries respectively for few indoor and outdoor scenes. This exercise demonstrates that the LiDAR data can be easily manipulated to deceive algorithms.

Next, we apply forensic algorithms *IsDensityConsistent* (I) and *IsOcclusionConsistent* (II) to 44 original scans and 66 forged scans. For algorithm *IsOcclusionConsistent*, we used 0.06 resolution factor for both θ and ϕ based on angle resolution of Riegl VS-400. As the forensic algorithms are designed focusing on additive attacks, we evaluate their performance on CTP and CRTP type forgeries only. Tables 6.1, 6.2 enlist the performance of the proposed forensic algorithms on the above mentioned dataset. It can be seen that the algorithm *IsDensityConsistent* works well for the CTP type forgeries, but most of the CRTP type of forgeries remain undetected. It can be observed from Figure 6.6 that, the MID increases as the distance from the sensor increases for original data. But the insertion of the high-density hoax object in the scene causes a sudden, persistent drop in it. Whereas re-sampling the hoax object to match the sampling density in the original scene before inserting it into the scene does not alter the MID distance and hence, remains undetected. On the other hand, the algorithm *IsOcclusionConsistent* performs well for both CTP and CRTP types of forgeries. As the algorithm *IsOcclusionConsistent* does not depend on the density, it even detects CRTP types forgeries with significantly high accuracy.

6.7 Discussion

Some observations made during the study include:

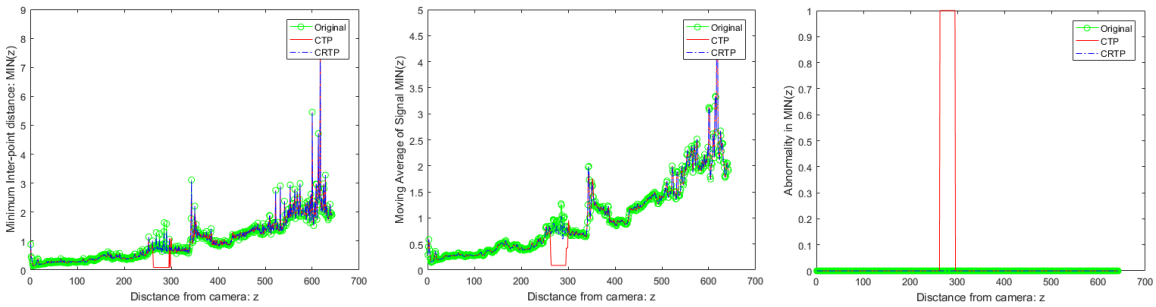
Implementation Complexity: Additive attacks are easy to implement as they only require the forged objects and their required orientation and placement in the scene. On the other hand, subtractive and deforming attacks need an additional understanding of the scene. Though point cloud segmentation can be used as a tool to understand the scene and identify the object of interest, it increases the complexity of the attack significantly. Further, given



(a) (b) (c)

(d) (e)

Figure 6.5: For LiDAR scan of the indoor scene 1 a) Original, b) CTP forged data, and c) CRTP forged data; Indoor scene 2 d) Original, and e) subtractive forgery.



(a) (b) (c)

Figure 6.6: a) $MIN_{ipd}(z)$, b) Moving average of signal $MIN_{ipd}(z)$, and c) Abnormalities/sudden changes in signal $MIN_{ipd}(z)$.

Table 6.1: Performance of *IsDensityConsistent* and *IsOcclusionConsistent* on the experimental dataset.

Outdoor Scenes	Number of Scenes	Detected as Original		Detected as Forged	
		I	**II**	**I**	**II**
Original	27	25	25	1	1
CTP	26	1	0	25	26
CRTP	14	13	1	1	13
Indoor Scenes	Number of Scenes	Detected as Original		Detected as Forged	
		I	**II**	**I**	**II**
Original	17	16	16	1	1
CTP	16	0	0	16	16
CRTP	10	9	0	1	10

Table 6.2: Classification accuracies of *IsDensityConsistent* and *IsOcclusionConsistent* on the experimental dataset.

	Original		CTP		CRTP		Overall	
	I	**II**	**I**	**II**	**I**	**II**	**I**	**II**
Accuracy (in %)	93.18	95.45	97.62	100.00	8.33	95.83	76.36	97.27

the required parameters for forgery attack, all types of forgeries can be created approximately in less than *50 milliseconds*.

Effectiveness of Attacks: Visualization based applications of LiDAR data involve the surface reconstruction from the raw LiDAR data using various meshing techniques. Hence, the visual effect of forgery will also depend on the meshing technique used. In that case, more rigorous user study is needed to evaluate the effectiveness of the attack. As we only focus on applications that utilize raw 3D LiDAR data, the effectiveness of attack is determined solely based on the possibility to perform it.

Limitations of the Forensic Approach: The proposed algorithm *IsDensityConsisten* is effective in the case of CTP type of forgeries whereas the algorithm *IsOcclusionConsisten*

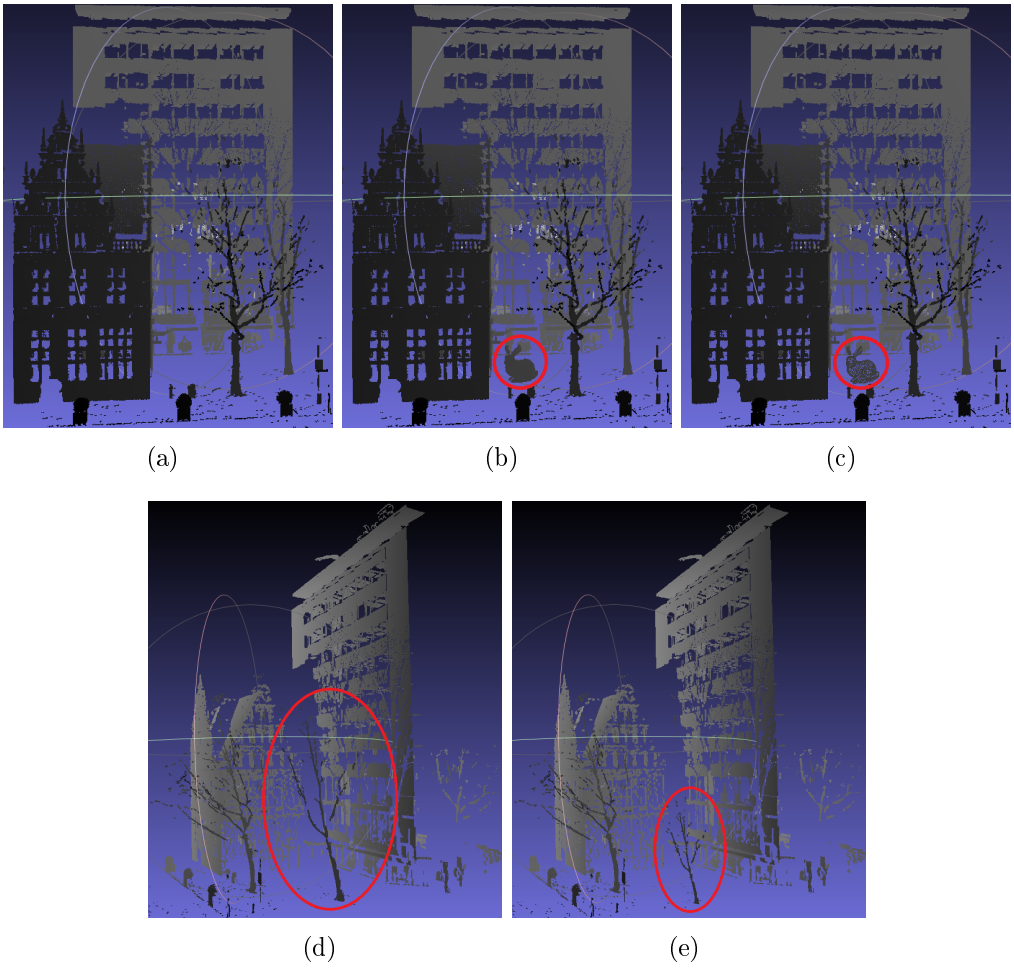


Figure 6.7: For LiDAR scan of the outdoor scene 1 a) Original, b) CTP forged data, and c) CRTP forged data; Side view of d) Original, and e) Deforming Forgery.

is efficient for both CTP and CRTP types of forgeries in LiDAR data obtained using a single scan. But, if the LiDAR data is obtained by fusing multiple scans, these approaches may not be useful for detecting forgery in such cases. More sophisticated forensic methods need to be investigated to detect the extensive set of forgeries including subtractive and deforming forgeries.

Fragile Watermarking: As described in Section 6.2, most of the existing watermarking methods are inappropriate to authenticate LiDAR data. One of the future approaches can be to design a real time fragile watermarking for LiDAR data.

CHAPTER 7

ALERT: ADDING A SECURE LAYER IN DECISION SUPPORT FOR ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS)[1]

7.1 Introduction

3D LiDAR (Light Imaging Detection and Ranging) sensors aid in sensing the environment where the distance is measured by projecting a laser beam on the target. Generally, LiDAR sensors use the *time of flight* approach for depth computation and the scanned data is stored as the list of points representing X, Y and Z coordinates (also termed as "unstructured 3D point cloud"). These sensors are typically accompanied with a pair of stereo cameras for obtaining the color information of the scene. LiDAR sensors outperform conventional color cameras due to their infrared signal based workflow which makes them useful at night time as well as in adverse environmental conditions. Therefore, most autonomous cars are equipped with LiDAR sensors.

However, considering the fatal crashes involving autonomous cars (McFarland, 2016, 2018) that occurred over a last couple of years, most of the self-driving car companies are adopting the concept of "teleoperation" to keep a human in the loop. Aligned with this concept, California State has recently amended the regulation for testing autonomous vehicles that require a human remote operator to continuously supervise vehicle's performance (State of California, 2018b) using a two-way communication link between a vehicle and the remote operator (State of California, 2018a). One of the self-driving companies, Drive.ai suggests that starting with one operator per vehicle, one tele-choice operator will eventually be able to monitor multiple vehicles at the same time (Ackerman, 2018). In such a scenario,

[1]

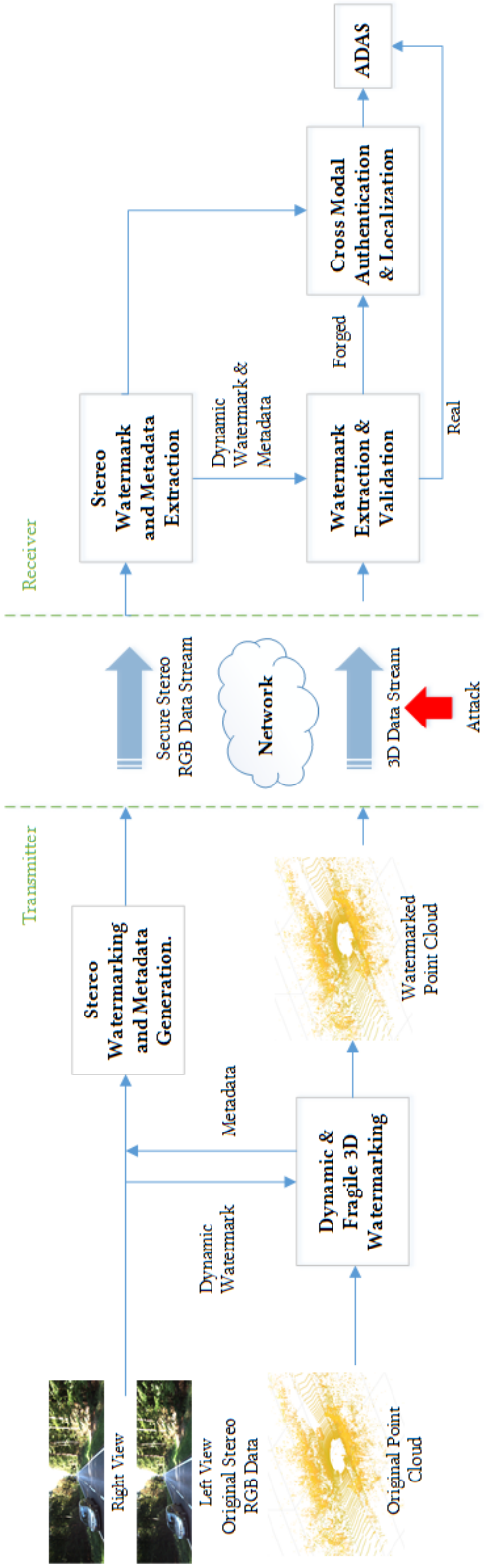


Figure 7.1: Overall workflow of the proposed ALERT framework.

it becomes difficult for a single operator to identify a forgery based on data consistency. Similarly, various mixed initiative remote robot control systems (Sauer, 2011; Heger and Singh, 2006) are introduced where a human can remotely guide a robot to complete a certain task without error.

In this context, as the data from a self-driving car or robot is communicated to a remote operator over a network, it is vulnerable to various forgery attacks. For example, the attacker may remove a vehicle in front of the self-driving car to create an illusion that there is no risk in increasing the speed and may result in a potential accident. Recently, (Petit et al., 2015; Petit and Shladover, 2015) showed that it is easy to stall a self-driving car just using simple commodity hardware. Thus, it becomes crucial to authenticate the data before utilizing it for decision making.

7.1.1 ALERT for ADAS

With this motivation, we propose a novel framework ALERT (Authentication, Localization, and Estimation of Risks and Threats) that adds a secure layer in the decision support systems used in Advanced Driver Assistance System (ADAS). Figure 7.1 describes an outline of the ALERT framework. As most autonomous cars use stereo RGB camera along with LiDAR, ALERT creates a dynamic watermark based on the stereo RGB data and employs fragile watermarking for tamper-proofing the 3D LiDAR data. The forgery detection in the proposed framework is a 2-fold setup that consists of a dynamic watermarking along with a cross-modal authentication and localization. It first identifies the attack by verifying the extracted watermark. If tampering is detected, the second step is triggered to localize the tampered region. ALERT also identifies the type of attacks and determines the risk factor associated with the data. While risk factor associated with an attack can be defined in different ways, the proposed formulation of the risk factor provides a holistic definition of the risk or threats based on various aspects of attacks including spatial and temporal consistency.

ALERT provides a complete set of information to ADAS to make a more informed decision. Additionally, it can notify the tele-choice operator about the possible risk which is beneficial when one tele-choice operator is monitoring multiple vehicles. Experimental results show the effectiveness of the ALERT framework in terms of: a) imperceptibility and efficiency of the proposed dynamic watermarking; b) overall execution time taken by various steps in the framework and c) accuracy of forgery detection based on the discrepancy in the watermark and cross-modal authentication and localization.

Assumptions: As described in Figure 7.1, we assume that the stereo data is secured with traditional watermarking techniques (Doerr and Dugelay, 2003; Liu and Zhao, 2010). We focus on attacks targeting 3D LiDAR data.

Contributions: To summarize the principal contributions:

- Efficient and effective ordering scheme that overcomes the challenges posed by lack of surface information and sparse nature of 3D LiDAR data for tamper-proofing it.
- Dynamic nature of watermark generation and embedding that enhances the security.
- Cross-modal authentication and risk factor assessment that analyze the 3D LiDAR data thoroughly.

7.2 Related work

3D data authentication and verification have been done in two ways: a) Forensic analysis of the characteristics of the data to detect forgery attacks, and b) Embedding a secret data as a watermark to verify the authenticity of the 3D data.

7.2.1 Forensics Analysis

Over the last decade, forgery detection for images and videos has been well-studied (Farid, 2009; Milani et al., 2012). Forensic analysis of 3D data, however, is relatively less explored

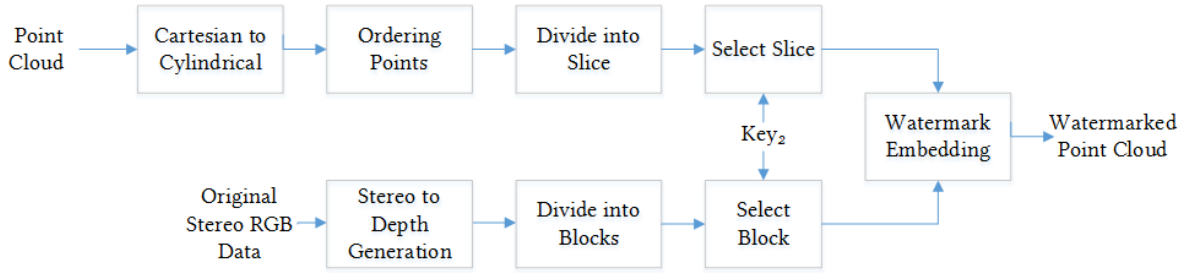
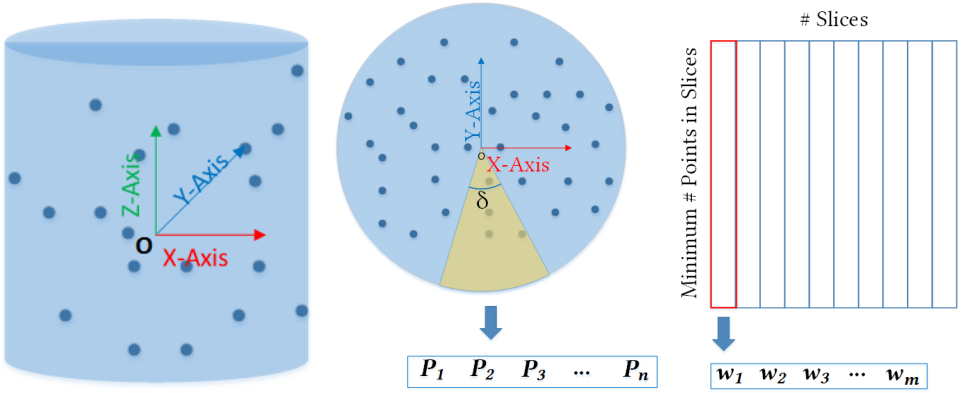


Figure 7.2: Real-time 3D dynamic watermarking.

with the exception of a few methods (Raghuraman et al., 2015b; Bahirat and Prabhakaran, 2017). Raghuraman et al. (Raghuraman et al., 2015b) designed an anti-forensic framework to capture and manipulate the live RGB-D data stream to create a realistic illusion of an individual performing the activities which they did not actually do. Presenting one of the key work in the area of 3D forensics, Bahirat et al. (Bahirat and Prabhakaran, 2017) identified various possible attacks on 3D LiDAR data as well as two forgery detection methods based on consistency in the key characteristics of LiDAR data such as: sampling density and occlusion effect.

7.2.2 3D Watermarking

Generally, watermarking methods can be broadly classified into two categories: robust and fragile methods. Robust methods (Wang et al., 2012; Jing et al., 2014; Yu et al., 2003; Zafeiriou et al., 2005; Praun et al., 1999; Yin et al., 2001) are designed to reconstruct the watermark even in case of malicious attacks for providing ownership protection. Whereas fragile methods (Yeo and Yeung, 1999a; Lin et al., 2005) are designed to detect even a very slight modification in a data for authentication purpose. In this paper, we will mainly focus on the 3D fragile watermarking.



(a) (b) (c)

Figure 7.3: a) Cylinder enclosing the 3D space, b) Top view and the selected slice (highlighted with yellow) along with the corresponding 1D vector, c) Blockwise layout of the watermark and 1D vector corresponding to the selected block (marked in red).

Fragile 3D Watermarking

Fragile watermarking methods should possess two desired properties: 1) vulnerability to even a slight modification of data; 2) localizing any kind of attack.

Spatial domain: Fragile techniques in the spatial domain generally modify the individual vertex positions to achieve vulnerability against different attacks. For example, methods (Yeo and Yeung, 1999b; Lin et al., 2005; Chou and Tseng, 2006) utilize an approach that identifies a new position for each vertex where two predefined hash functions have an identical value, in order to make all vertices valid for authentication (Wang et al., 2008). However, these approaches suffer from causality problem (please refer to (Wang et al., 2008)) and need 1-ring neighbor information. Cayre and Macq (Cayre and Macq, 2003) proposed a blind data-hiding algorithm for triangular meshes where the projection of a vertex on its opposite edge of the triangle is selected as a primitive for watermarking.

Transform domain: To achieve imperceptibility, researchers often embed the watermark in a spectral domain based on the 3D mesh multiresolution analysis. Chao et al. (Cho et al., 2004) proposed a fragile method that performs several decompositions of original tri-

angular mesh and utilizes facets in the coarser mesh as authentication primitive. Wang et al (Lounsbery et al., 1997) described another approach where after one wavelet decomposition, the norm and orientation of each wavelet coefficient vector are modified independently.

Point cloud watermarking: Among the early work, Ohbuchi (Ohbuchi et al., 2004) generated a non-manifold mesh from a point cloud and performed a mesh spectral analysis for embedding watermark into spectral coefficients. Cotting (Cotting et al., 2004) designed a technique that: (a) partitions a point cloud data into patches; (b) transforms them into discrete frequency bands; (c) embeds a watermark into the low-frequency components. Aggarwal et al. (Agarwal and Prabhakaran, 2007) proposed a blind watermarking that uses a quantization index of bit encoding to embed watermark bits into ordered points.

Limitations and Challenges: Most of the algorithms described are designed for 3D polygonal meshes using topological information, connectivity, and surface parameterization. Since LiDAR data do not have surface and connectivity information associated with it, we can not use most of the 3D mesh watermarking methods. Unlike existing fragile watermarking methods, the proposed watermarking is robust against re-ordering of points that is likely to occur during network transmission.

7.3 ALERT: Authentication, Localization, and Estimation of Risks and Threats

Most autonomous vehicles (and robots) are equipped with a LiDAR sensor as well as a pair of stereo color cameras. We propose a novel framework ALERT (Figure 7.1) that adds a secure layer for a navigation system that uses both LiDAR and stereo RGB data. It consists of three building blocks:

- A dynamic watermark based on stereo RGB data and embed it in 3D LiDAR data using fragile watermarking techniques.
- Cross-modal authentication and localization
- Risk factor computation

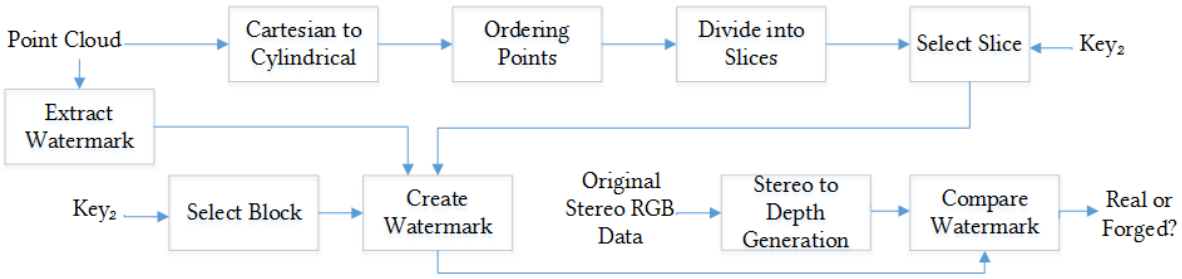


Figure 7.4: Watermark extraction and validation.

7.3.1 Dynamic 3D Watermarking

For achieving the desired tamper proofing of 3D LiDAR data, the proposed watermarking scheme (Figure 7.2) must possess the following key characteristics: a) **Dynamic watermark** is obtained by creating a depth map from the stereo data. However, to correctly embed a 2D watermark into a list of 3D points, each 3D point must be mapped to a unique pixel in the 2D watermark. Such mapping is achieved by ordering 3D points as described later in this section. The additional benefit of the ordering strategy is the robustness against shuffling the points. Such a robustness against shuffling is advisable as merely re-ordering the 3D points does not affect the scene understanding and consequently the navigational decisions; b) **Imperceptibility** in embedding is attained by restricting the effect of embedding the watermark to the least significant bits (LSBs) of each 3D point; c) **Fragility** of the watermarking is very important to realize tamper-proofing of the data which is achieved by spreading the watermark across all 3D points. Such that any kind of tampering is easily reflected in the extracted watermark.

Watermark Generation:

For increased security, instead of using a static watermark, we suggest utilizing a dynamic watermark that changes for every frame. We obtain the dynamic watermark by extracting the binarized depth map from the stereo data. To construct a depth map, first a disparity

map from stereo data can be obtained using traditional stereo correspondence (Olofsson, 2010), followed by depth computation using a pin-hole camera model.

Watermark Embedding:

The complete watermark embedding process consists of various steps:

Ordering 3D points: helps to create a mapping such that each 3D point is mapped to a unique pixel in a 2D watermark image. First, all the points are mapped to a Cylindrical coordinate system with the origin located at the position of the LiDAR sensor and z-axis aligned with the z-axis of the LiDAR (see Figure 7.3 a). Mathematically, the corresponding mapping for a 3D point (x, y, z) in the Cartesian coordinate system to the corresponding (r, θ, z) in the Cylindrical coordinate system is given by:$r = \sqrt{x^2 + y^2 + z^2}$, $\theta = \arctan(\frac{y}{x})$, $z = z$; where r, θ and z are radial, azimuth and height coordinates.

All the points represented in the Cylindrical coordinate system are then sorted based on the θ value and divided into different slices with angular width δ as shown in Figure 7.3 b. Hence, the total number of slices will be, $N_{slice} = \frac{360}{\delta}$. For each slice, all the corresponding points are first sorted with respect to z followed by r in the ascending order. This will form a 1D vector of all the 3D points in that slice (see Figure 7.3 b). The generated watermark is then divided into N_{slice} vertical blocks (see Figure 7.3 c).

Slice and Block Selection: As there are N_{slice} number of 3D data slices as well as watermark blocks, there is a one to one mapping possible between them. However, this correspondence between slices and blocks is randomized for enhanced security. Based on the predetermined key (key_2), slice and the blocks are randomly chosen for embedding the watermark information.

Embedding: the watermark block into the corresponding slice depends on two factors: a) Watermark image size and b) Number of points in the slice, K. The size of the cropped watermarked image is selected based on the statistical analysis of the experimental dataset

that determines the minimum number of points M in any slice. In the current experimental setup, the watermark image size is empirically determined to be $M \times N_{slice}$. Based on values of M and K various conditions are possible. When $M = K$, each watermark bit is simply embedded in the corresponding 3D point. If $M < K$, then a K points are divided into groups where each group contains M points except the last group. M bit watermark is replicated randomly across all groups. However, the last group with $< M$ number of points will contain only partial watermark information. If $M > K$ for any slice, it is merged with the next smallest slice. This process is repeated until the smallest slice has $K > M$. The information about slice merging is provided as a metadata at the receiver side. Please note that N_{slice} will also be updated based on merging operations.

Embedding Strategies:

For embedding watermarking information in 3D points, one can use any traditional embedding strategies such as:

Low Bit Modulation (LBM): Due to simplicity, Low bit modulation is widely used in various watermarking techniques. In this strategy, the watermarked data is simply stored in the LSB (Least Significant Bit) of each 3D point which makes it imperceptible with least perturbation. To make sure the LSB only represents the watermark data, we first set the LSB of each 3D point to zero followed by replacing the LSB of each point with the corresponding watermark bit. Similarly on the receiver side, first the information stored in the LSB of the point cloud data is extracted. This LSB bit is then set to zero followed by the proposed ordering scheme. The mapping between 3D points and 2D pixels is then utilized to construct the extracted watermark image.

Quantization Index Modulation (QIM): QIM is a popular embedding strategy as it provides a good trade-off between imperceptibility, watermark capacity and robustness (Chen and Wornell, 2001). We adopt the basic QIM strategy proposed in (Chen and Wornell, 2001)

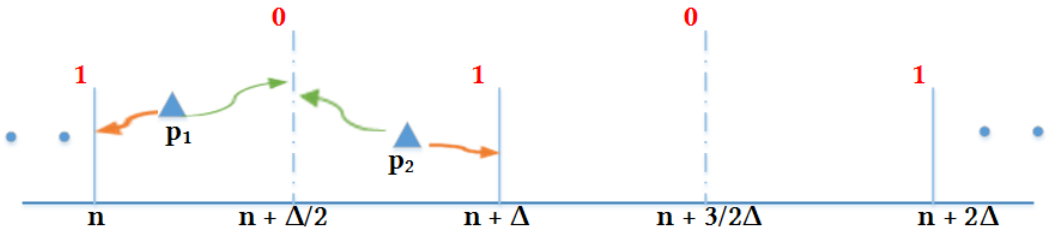


Figure 7.5: Quantization index modulation. Each point is quantized to nearest reconstruction point for "0" or "1" bit.

with slightly modified quantizers with quantization step Δ. To embed the watermark bit "1" in "z" coordinate of point p, we quantize it by

$$Q_1(z) = \Delta\lfloor\frac{z}{\Delta} + \frac{1}{2}\rfloor. \tag{7.1}$$

To embed the watermark bit "0", we quantize the z coordinate by Q_0

If $z < Q_1$

$$Q_0(z) = \Delta\lfloor\frac{z}{\Delta} + \frac{1}{2}\rfloor - \frac{\Delta}{2}. \tag{7.2}$$

Else,

$$Q_0(z) = \Delta\lfloor\frac{z}{\Delta} + \frac{1}{2}\rfloor + \frac{\Delta}{2}. \tag{7.3}$$

Figure 7.5 illustrates the QIM in the case where one bit (0 or 1) is embedded per point. Thus, there will be two quantizers and the corresponding set of construction points in $\Re$ are shown in Figure 7.5. So if we want to embed bit "1", the point p_1 will be quantized to n and to embed "0" it will be quantized to $(n + \frac{\Delta}{2})$.

At the receiver side, for extracting the watermark bit $wm_{extract}$ embedded in point $p(x, y, z)$, we solve for following equation:

$$wm_{extract}(p) = \underset{i=0,1}{\operatorname{argmin}}(Q_i(p(z)) - p(z)) \tag{7.4}$$

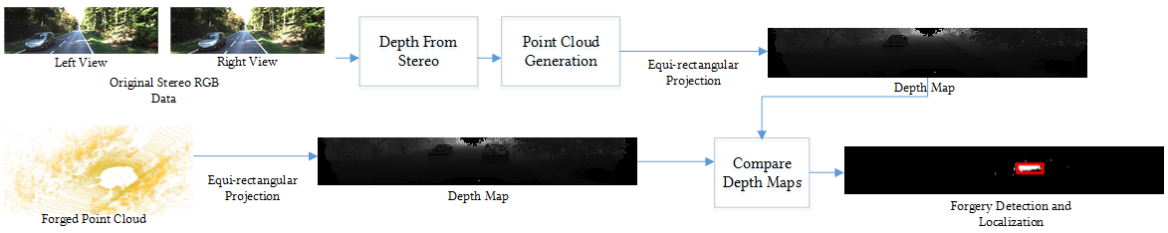
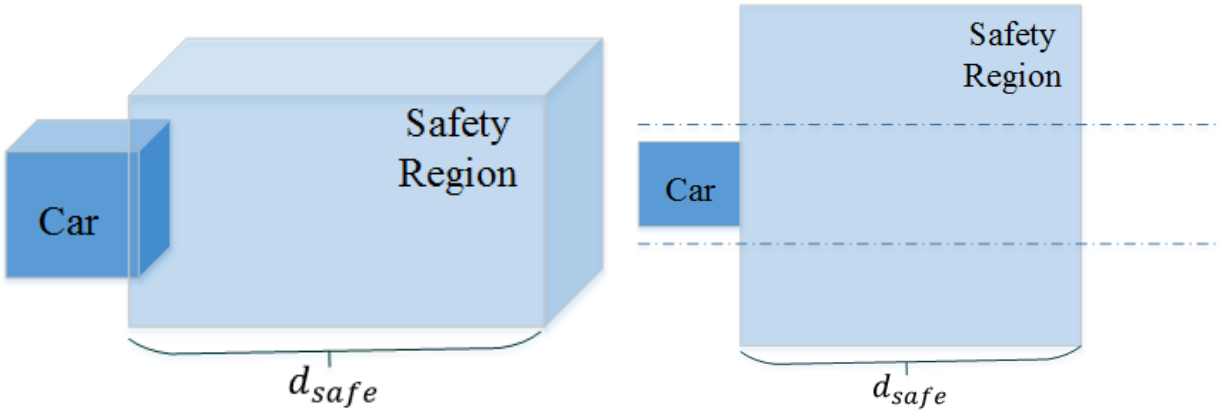
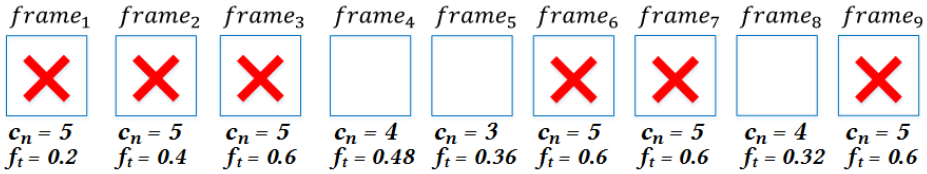


Figure 7.6: Cross-modal authentication and localization.

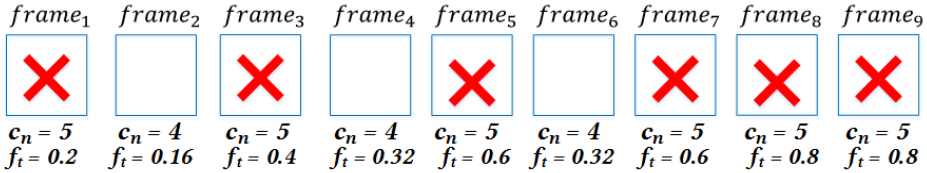


(a) (b)

Figure 7.7: Safety Region Illustration a) Side view, b) Top View (d_{safe} is the minimum stopping distance at current speed).



(a)



(b)

Figure 7.8: Visualization for the temporal element of the Risk factor with $X_n = 5$ in case of a) Pattern 1, b) Pattern 2.

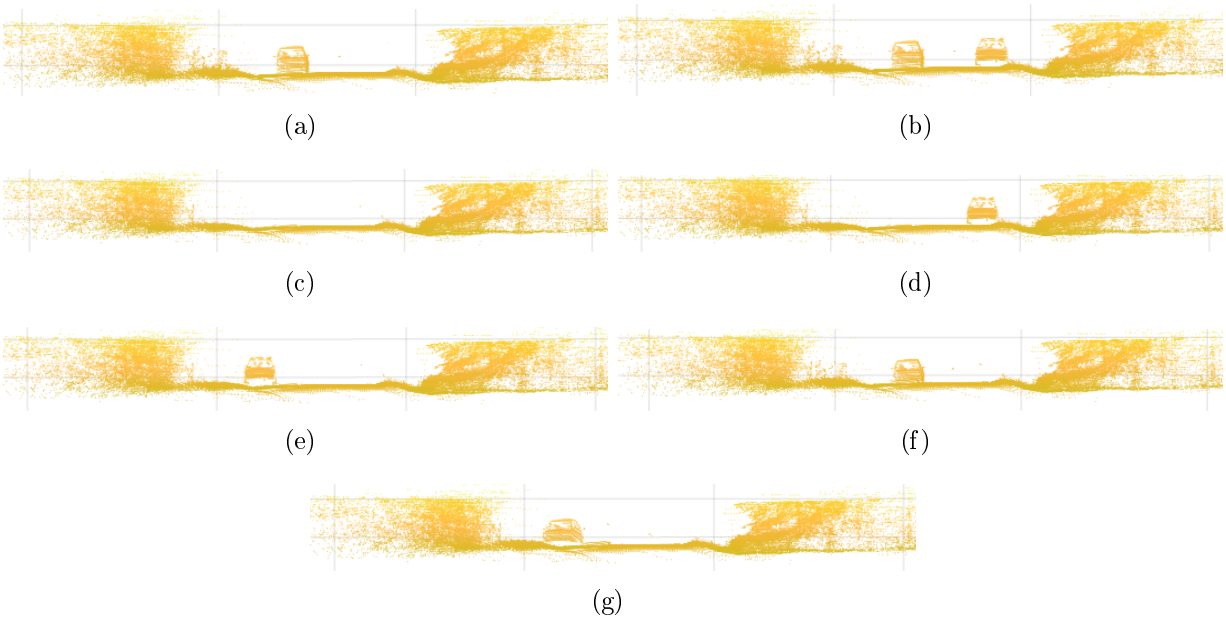


Figure 7.9: Illustration of different types of attacks on frame 6 from Drive 1: a) Original point cloud, b-g) With attack 1-6.

Meta data communication:

For a successful operation of the proposed watermarking method, a metadata consisting of keys for slice and block selection, slice merging information and the size of watermark must be communicated to the receiver end via a secured channel. We suggest embedding this information in stereo data via a traditional image watermarking. To achieve this, the image watermarked must have additional capacity to store the metadata while satisfying the real-time constraints described later.

Watermark Extraction and Validation:

Figure 7.4 describes a watermark extraction and validation process carried out at the receiver end of the system. First, the watermark information is extracted from the point cloud and saved as $wm_{extract}$. All the points in the received point cloud are mapped to a Cylindrical coordinate system and ordered as described in Section 7.3.1. The ordered set of points are

divided into slices with angular width δ each. Using the predefined key (key_2) and metadata, the desired slice and corresponding block is selected to create a 2D binary watermark image from the extracted watermark data $wm_{extract}$. Simultaneously, a dynamic watermark is re-generated at the receiver end using stereo data. These two watermarks are then compared pixelwise to determine if the data is real or forged. Please note that, for the slices where $M < N$, the number of extracted bits are divided into groups of M bits and logical AND operation is performed to compute the watermark information in the corresponding block.

7.3.2 Cross-modal Authentication and Localization

As the same scene is observed by two modalities: LiDAR sensor and stereo cameras, performing a cross-modal authentication is a natural choice where the data captured using one sensor can be validated using the data from another sensor. However, the data captured using these two modalities are in different domains. The stereo data is a set of 2D images while the point cloud is a set of 3D points. To bring them to the same domain for comparison, we process them as described in Figure 7.6. First, we generate a depth map from the stereo data and project it to 3D using the camera parameters. It will generate a corresponding point cloud data. As the estimated depth maps from stereo images can be noisy, we can not directly perform a point to point comparison between stereo and LiDAR point cloud. Hence, we apply an equirectangular projection to get a 2D depth map as shown in Figure 7.6. Similarly, we also apply equirectangular projection to the LiDAR point cloud. Now, as both the modalities are in same domain, we can verify if the objects present in one modality are also there in another modality. Due to the 2D representation of data, we can easily identify the modified region using connected component analysis. As mentioned earlier, we assume that the stereo data has been authenticated using traditional video/image watermarking techniques.

7.3.3 Risk Factor Computation

The decision making in autonomous vehicles is a complex system and involves multiple sensors. Such systems account for the fact that the performance of a sensor is highly dependent on the environmental conditions and sensor's reliability and precision might be affected by unfavorable conditions. Thus, for reliability, these systems need the self-assessment of each sensor where each sensor notes its own limitations and signals this information to the control unit (Stiller et al., 2000). Being consistent with existing decision-making systems, we design a "Risk Factor" associated with the LiDAR data that conveys the reliability and risk associated with the data.

Needless to say, the risk factor can be defined in different ways. In this paper, we propose a risk factor R_f ranging between (0,1) where 0 indicates minimum risk while 1 indicates the highest risk. We consider a region (referred to as "safety region") in front of the car that is more critical. The dimension of a safety region depends on the minimum stopping distance at current speed d_{safe}, the height of the car and width of lanes. Figure 7.7 illustrates the safety region and its coverage. To define the risk factor R_f, we mainly consider the following six key elements:

A distance of the attacked region from the vehicle, d. If the distance is d_{safe}, the risk is lower as we can probably stop before the attacked region. Hence, the corresponding factor is defined as $f_d = max(0, \frac{(d_{safe} - d)}{d_{safe}})$.

The number of attacked points, n. Higher the density of attacked points, higher is the risk. Thus, the corresponding factor is $f_n = \frac{n}{N}$, where N is the total number of points in the safety region.

The type of attack, α. Different types of attack may pose different amount of risk. So, we define the corresponding factor as $f_\alpha = \alpha$, α can be between (0,1) based on the type of attack.

Angle of the attacked region with the forward direction of the vehicle, θ helps to identify if the attacked region is in the current lane or side lanes. $theta = 0$ indicates that the attack is in the current lane which introduces higher risk. Hence, we formulate the corresponding factor as $f_\theta = cos(\theta)$

Spatial correlation of the attack, s**.** If the same region is attacked in two consecutive frames, the d and θ will be probably same across these frames. Hence to capture the spatial correlation, the corresponding factor is defined as $f_s = \text{average}_{i=0,X_n}(corr(s_i, s_{i-1}))$, where X_n is the total number of previous frames and $s_i = [d_i, \theta_i]$.

Temporal consistency of the attack, t**.** It includes two sub-factors: a) the percentage of frames being attacked and b) history information signifying the steadiness of the attack. The corresponding factor is $f_t = \frac{x_n}{X_n} \times \frac{c_n}{X_n}$, where x_n are the number of frames that are attacked, c_n holds the history information indicating the steadiness of attack. If the frame is attacked, we set $c_n = X_n$. But if there is no attack, it is updated as $c_n = c_n - 1$ until it reaches to 0. Among these elements, the first six elements are based on information in the current frame only, while spatial and temporal elements take a set of previous frames into account.

Definition: Based on the above parameters, risk factor R_f is defined as:

If the attack is detected, $R_f = f_d \times f_n \times f_\theta \times f_\alpha \times f_s \times f_t$

If no attack is detected, $R_f = f_s \times f_t$

For computing the temporal and spatial element of the risk factor, we consider X_n number of the previous frame. X_n is computed based on the current speed of the vehicle and time it would take to cover the distance of 100 meters (the distance for which one would like to observe the attack). Figure 7.8 illustrates the computation of c_n and f_t in various attacks. This formulation helps to distinguish between the attack patterns with the same number of attacked frames x_n but different attack distribution as shown in Figure 7.8.

7.4 Attacks

To evaluate the vulnerability of 3D LiDAR data, we consider 6 different possible attacks (See Figure 7.9) by an attacker trying to deceive the decision-making module in autonomous system.

Additive attacks (Attack 1): The attacker can introduce a new object to suddenly stop the autonomous vehicle (Figure 7.9 b). To perform such attacks, the region of interest in one point cloud is extracted and inserted into another by simply appending points to the initial list.

Subtractive Attack (Attack 2): As the objects that are near and in front of the LiDAR sensors are very important, we identify points in proximity of LiDAR sensor. These points are then removed to perform a subtraction attack (see Figure 7.9 c).

Simultaneous Additive and Subtractive attack (Attack 3): To increase the complexity of attack, we perform both additive attack and subtractive attack together (refer to Figure 7.9 d).

Replacement attack (Attack 4): In this case, the forged object is added in the scene to replace an existing object (Figure 7.9 e).

Translation (Attack 5): To trick the decision support module, the attacker may want to move the object a little farther to change the decision (Figure 7.9 f). We achieve it by identifying the nearest object and translate it away from the LiDAR sensor.

Rotation (Attack 6): The attacker may want to rotate the car parked on the one side of the road to alter the decision (Figure 7.9 g). This is achieved by identifying the nearest object, computing the center of mass and translate it to the origin, rotate it about y-axis by a predefined angle and translate it back to its original position.

Attack Type Detection: All the attacks described earlier have different levels of risk associated with them. For example, the attack where the car in front is forgedly remove with a subtractive attack is more dangerous than the additive attack. Thus, determining

the type of attack is crucial for determining the risk factor described in the earlier section. We broadly classified all the above attack into three categories: additive, subtractive and modification attack. All the attack 4-6 are termed as modification attack. To determine the type of attack the depth map obtained from the LiDAR data is masked with the difference image obtained using a cross-modal module. Finally, the type of the attack is determined based on the depth values in the masked region and corresponding area. For additive attack, the masked area will have non-zero depth values in the depth map. While, for subtractive and modification attack, the masked region will have a blob of pixels with zero depth values. However, the area of the zero depth patch will significantly differ in the case of subtractive and modification attack.

7.5 Experimental Results

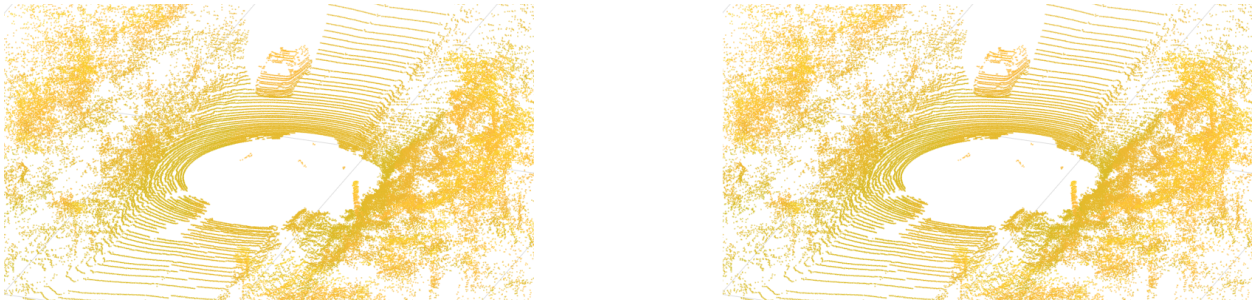
In this section, we demonstrate the efficiency of the ALERT framework. Mainly, we carried out 4 different set of experiments that: a) Demonstrate the imperceptibility of the proposed fragile watermarking method, b) Exhibit the efficiency of the proposed watermarking approach by computing a recovery rate, c) Verify the real-time performance by enlisting the execution time taken by various steps and d) Evaluate the accuracy of both the proposed fragile watermarking as well as cross-modal module under various attacks described in Section 7.4. All the attacks and algorithms are implemented in MATLAB and experiments are run on a CPU with Intel (R) Core (TM) i7-5820K with 3.30GHz speed and 32GB internal RAM. Various algorithm parameters used in the experiments are: a) With the statistical analysis of the experimental dataset (described later), the minimum number of points in any slice M is empirically determined as 1000; b) To get significant number of slices, angular width δ is set to 10; c) To keep the distortion level minimum due to watermark embedding, we use step size $\Delta = 0.0025$.

7.5.1 Dataset

We utilize LiDAR and stereo data provided by the KITTI Vision Benchmark Suite (Geiger et al., 2013). It consists of a variety of environments such as City, Residential, Road etc scanned using multiple sensors (please refer to (Geiger et al., 2013)). In our experiments, we mainly considered the 3D point clouds scanned using Velodyne HDL-64E and stereo color images captured using 1.4 Megapixels Point Grey Flea 2 cameras. *Please note that the point cloud data and stereo data are synchronized and rectified.* We primarily focused on the drives in the "Residential" area as there are multiple cars parked on both sides of the road that allow us to create a significant number of attacks of type 2-6. The experimental dataset consists of three drives captured on 09-26-2011: drive 22, 39 and 64 (referred as Drive 1, 2 and 3, respectively) with 800, 395 and 570 frames each. Table 7.2 lists all the attacks performed to create forged data as described in Section 7.4. The total number of attacks created for Drive 1, Drive 2 and Drive 3 are 4135, 2130 and 2470 respectively. So overall in this experimental setup, we evaluated 8735 different types of attacks.

7.5.2 Evaluation

Imperceptibility. One of the key characteristics of the watermarking technique is the imperceptibility that requires the watermark embedding to introduce the least amount of distortion in the data. In the proposed approach, the desired imperceptibility is achieved by restricting the effect of embedding to the least significant bits of each point in the data. Figure 7.10 demonstrates the imperceptibility achieved using the proposed watermarking with QIM embedding. To quantitatively evaluate the imperceptibility, we compute a Hausdorff distance as described in (Cignoni et al., 1998) between the original point cloud and the watermarked point cloud for all the frames in each drive. The average Hausdorff distance across all drive is 0.001732 with LBM and 0.00125 using QIM.



(a) (b)

Figure 7.10: Illustration of imperceptibility of the proposed watermarking: a) Original point cloud, b) Watermarked point cloud (with QIM embedding).

Efficiency. To compute the efficiency of the proposed fragile watermarking method, we embed and extract the watermark with both the embedding strategies (LBM and QIM) under no attack condition. The recovery rate is recorded for every frame in each drive. The proposed method demonstrates 100% recovery rate in case of all the drives included in the experimental dataset.

Execution time. In the KITTI Vision Benchmark Suite (Geiger et al., 2013), the LiDAR data is typically scanned at 10 fps while the synchronized and rectified stereo data is available only at 1 fps. Thus, we have 100 milliseconds (ms) to process each LiDAR frame for watermark embedding and extraction. Similarly, we have approximately 1 second for performing cross-modal authentication. The "real-time" aspect of performance is defined based on the aforementioned time constraints. To evaluate the performance of the proposed system in terms of execution time, we measure the average execution time taken by different steps of the proposed framework for each drive. As can be seen from Table 7.1, we can process each frame for watermark embedding and extraction in approximately 100 ms each. Similarly, the execution time taken by cross-modal authentication module is approximately 200 ms satisfying the required time constraints. Please note that the reported times do not include watermark generation as it is considered as a pre-processing step.

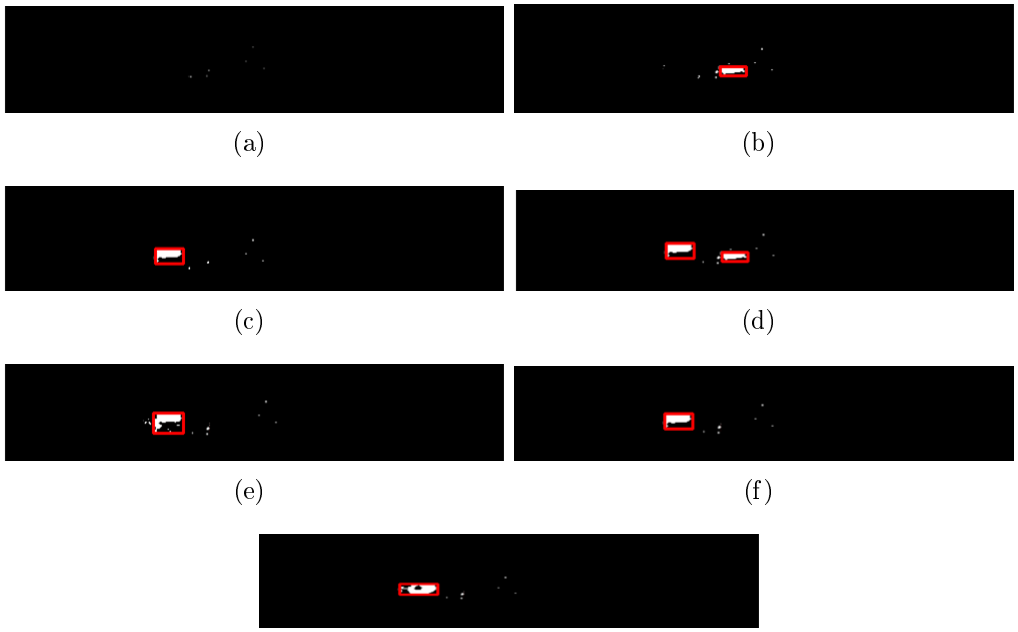
Table 7.1: Execution time taken by various steps of ALERT.

	Average Execution time (in ms)		
	Drive 1	Drive 2	Drive 3
Ordering Points (a)	61.80	59.10	59.40
LBM Embedding (b)	38.00	35.80	36.80
QIM Embedding (c)	48.80	47.90	49.10
Execution time with LBM at Tx ($a+b$)	**99.80**	**94.94**	**96.20**
Execution time with QIM at Tx ($a+c$)	**110.60**	**106.00**	**108.50**
LBM Extraction (d)	47.50	45.40	47.20
QIM Extraction (e)	32.90	31.80	32.80
Watermark Comparison (f)	0.08	0.07	0.05
Execution time with LBM at Rx ($a+d+f$)	**109.38**	**104.57**	**106.65**
Execution time with QIM at Rx ($a+e+f$)	**94.78**	**90.97**	**92.25**
Depth From stereo (g)	172.98	174.12	173.24
Rectilinear projection (h)	12.35	12.21	12.45
Depth Comparison (i)	4.07	4.25	4.14
Execution time for cross-modal ($g+2\times h+i$)	**201.75**	**202.79**	**202.28**

Accuracy. The proposed framework is evaluated by analyzing the accuracy of: a) the proposed fragile watermarking, and b) cross-modal authentication and localization. The original point cloud data is watermarked using the proposed watermarking with both embedding strategies (LBM and QIM). The data then undergoes various attacks listed in Section 7.4. The attacked data is processed through the step of watermark extraction and cross-modal module. Any discrepancy in the extracted watermark is detected as a forgery. Similarly, any difference between the LiDAR data and the stereo data is captured as forged data by a cross-modal module. Figure 7.12 illustrates the performance of the proposed cross-modal module in localizing various types of attacks. Table 7.2 shows the statistics of the attack detection for the proposed watermarking with LBM and QIM embedding strategy and for the proposed cross-modal authentication and localization. Overall, the proposed dynamic

(a) (b)

Figure 7.11: Failure case of cross-modal authentication: a) Depth map from LiDAR data, b) Depth map from stereo data (noisy region marked with the red rectangle).



(a) (b) (c) (d) (e) (f) (g)

Figure 7.12: Performance of cross-modal authentication and localization in case of: a) Original point cloud, b-g) With attack 1-6.

watermarking achieves 100.00% accuracy with both LBM and QIM embedding. The cross-modal module achieves 97.56% of overall accuracy. The reduced accuracy with a cross-modal module is mainly due to noisy depth estimation from stereo data as shown in Figure 7.11. Additional results are provided in the **supplementary material**.

Table 7.2: Statistics of attack detection for the proposed watermarking with LBM and QIM embedding strategy and for cross-modal authentication and localization.

Type	Drive 1							Drive 2							Drive 3						
		LBM		QIM		Cross-modal			LBM		QIM		Cross-modal			LBM		QIM		Cross-modal	
	# of Frame	O	F	O	F	O	F	# of Frame	O	F	O	F	O	F	# of Frame	O	F	O	F	O	F
Original	800	800	0	800	0	772	28	395	395	0	395	0	382	13	570	570	0	570	0	564	6
Attack 1	800	0	800	0	800	11	789	395	0	395	0	395	1	394	570	0	570	0	570	0	570
Attack 2	667	0	667	0	667	25	642	347	0	347	0	347	12	335	380	0	380	0	380	11	369
Attack 3	667	0	667	0	667	0	667	347	0	347	0	347	2	345	380	0	380	0	380	1	379
Attack 4	667	0	667	0	667	20	647	347	0	347	0	347	1	346	380	0	380	0	380	0	380
Attack 5	667	0	667	0	667	24	643	347	0	347	0	347	9	338	380	0	380	0	380	13	367
Attack 6	667	0	667	0	667	43	624	347	0	347	0	347	16	331	380	0	380	0	380	20	360
Precision(%)		100.00		100.00		99.31		-	100.00		100.00		99.38		-	100.00		100.00		99.75	
Recall(%)		100.00		100.00		97.03		-	100.00		100.00		98.08		-	100.00		100.00		98.18	
Accuracy(%)		100.00		100.00		96.94		-	100.00		100.00		97.86		-	100.00		100.00		97.56	

Column “O” refers to detected as Original and “F” refers to detected as Forged.

7.6 Discussion

Dynamic watermark: As the algorithm is currently implemented on CPU in MATLAB, without any optimization, the execution time needed for the depth map generation from the stereo is approximately 170 milliseconds. GPU based implementation is very likely to reduce this execution time significantly. We can also consider generating a dynamic watermark for a LiDAR frame at time t in two other ways: a) using the depth map obtained from stereo data at time $t-1$ and b) using a 2D random noise pattern.

A trade-off between accuracy and time for depth from stereo: As described in Table 7.1, generating the depth map from stereo image pair is computationally expensive. If the depth map generation step needs to be expedited, it results in inaccurate depth estimates which consequently results in increased false positives during cross-modal authentication and localization. Thus, it becomes crucial to achieve a trade-off between the accuracy and execution time for depth map generation from stereo data.

Limited localization: The current dataset includes the stereo data capturing only the frontal environment. Thus, based on the data availability, the current approach is limited to localize attacks occurring in the frontal 180° only. However, the full 360° localization can be obtained if multiple stereo camera pairs are installed for scanning the complete environment.

Risk factor assessment: Moral machine experiment described in (Awad et al., 2018) demonstrates a substantial subjectivity and cultural variation associated with ethical judgments made by humans. Thus, the problem of modelling the risk factor with global consensus becomes more challenging. Hence, the risk factor can be customized by researchers in different ways as per the requirements.

Decision making: As mentioned earlier, the proposed framework assist in autonomous navigation of self-driving cars and robots. It provides a complete set of information including whether the data has been attacked or not, attacked region, type of attacks and risk factor associated with the data. This complete set of information is provided to ADAS for making

a more informed decision. For example, if the risk factor associated with the data is "1" and the attacked region is localized in the front of the vehicle and the side lane is marked as safe, then one can decide to pull over and stop. Although such decisions are subjected to the controversial debate of "what is an ethical decision" (Gent, 2017), it is important to provide some preliminary decision for a reliable navigational system. Also for cases where one tele-choice operator is supervising multiple autonomous vehicles, ALERT can notify the tele-choice operator regarding the potential risk.

CHAPTER 8

CONCLUSION

With the current advances in depth sensing technologies, the 3D content can be easily generated and is widely available in different formats. However, employing the 3D content effectively and reliably has several challenges. This dissertation addresses three main challenges by providing a set of novel algorithms. These algorithms assist in accelerating the effective and reliable utilization of the 3D content in a diverse set of applications and demonstrate future prospects of 3D technologies.

8.1 Challenges

Mainly we have addressed three challenges:

1. Limited processing power and resources in mobile HMDs: Mobile HMDs are severely constrained with low-processing power and limited onboard memory. Therefore, they can not handle the same virtual environments and objects as desktop-based VR systems without rendering at lower framerates, which can potentially induce the simulator sickness and degrades the user experience.

2. Effective usage of 3D content in virtual therapies: Although mixed and virtual reality techniques are increasingly employed in the healthcare domain, a detailed analysis of deploying them in specific task needs to be addressed. Considering a specific task of using mixed/virtual reality for managing phantom pain, existing virtual reality-based solutions are invasive in nature and rely on pre-built 3D models of the phantom limb. Such pre-built 3D models degrade the immersive experience due to a mismatch in the skin color, clothes, artificial and rigid look and misalignment.

3. A vulnerability of 3D content: Previous two challenges demonstrate the requirement of manipulating 3D content for useful tasks. However, the ability to easily manipulate

the 3D content exposes its vulnerability to various security threats. The concerns regarding the security risk associated with 3D data become more prominent in the background of its applications in various sensitive areas such as surveillance, autonomous vehicle control etc.

8.2 Proposed methods

To address the above-mentioned challenges, we propose a set of novel methods for 3D content manipulation with three different perspectives:

1. Simplification for multi-platform adaption: We opt for an automated mesh and surface simplification approaches to accommodate low processing power of mobile HMDs. Such that the 3D data can be adaptively simplified to make it deployable across various platforms ranging for low-power mobile head-mounted displays to desktops equipped with high power GPUs. We address the problem of multi-platform adaptation for different types of 3D content format independently:

Offline approach for 3D mesh simplification: We have described a new high-fidelity mesh simplification method, QEM_{4VR}, tailored for VR applications to yield low-poly meshes for any 3D model (both manifolds and non-manifolds). QEM_{4VR} method has two significant advantages:

- Its resulting meshes have a less geometric error than previous variations of the QEM simplification algorithm due to its curvature-based boundary preservation approach.
- Also, our QEM_{4VR} method is more capable of preserving surface properties, such as multiple texture coordinates for a single vertex, than previous approaches that attempt to preserve such surface properties.

We validated both key contributions through a series of experiments that compared our method to prior QEM variations.

Our perceptual quality assessment experiment shows that state-of-the-art perceptual quality metrics are incapable of adequately measuring the visual quality of simplified non-manifold meshes. It motivates the need to further explore and design a quality metric correlated with the human perception to evaluate mesh simplification methods. An adequate perceptual measure should incorporate boundary gaps, completeness of the model, and associated texture information while computing the error.

Real-time approach for surface simplification from depth images: Deriving the motivation from the challenges posed by the processing limitations of handheld VR devices and rendering engines in real-time immersive applications, we have presented a novel depth image based surface simplification method that utilizes the underlying grid structure of the depth image. The proposed CS^3 (Curvature Sensitive Surface Simplification) operator selects points based on their importance of curvature and hence is capable of preserving finer details of the surface even at the high level of sparse sampling. Modified constraint Delaunay triangulation makes the proposed method suitable for standalone models as well as complete scenes. Also, the proposed CS^3 method is capable of preserving surface properties such as texture coordinates. Fast execution time and a reasonable approximation to the underlying surface, make the proposed method suitable for real-time immersive applications. The proposed method generates a sparse mesh of a user-specified vertex count to mitigate system limitations. Various experimental results have demonstrated that the proposed approach maintains an acceptable level of visual quality even after significant data size reduction. The proposed method can be easily extended to obtain distance-dependent simplification. It can be used in bandwidth adaptive tele-immersion, level-of-detail rendering, 3D data compression.

2. Modification for virtual therapies: We have presented a novel Mixed Reality based system for MAnaging Phantom Pain (Mr.MAPP) that creates a 3D model of the person and immerse it in the interactive game environment. It identifies and removes the 3D

points corresponding to the missing limb in the real-time. It also provides a real-time phantom limb generation with the skeleton modification for accurate movement of the phantom limb and its interaction with the gaming environment. The Mr.MAPP framework bestows a cost-effective, simple and non-invasive solution for the relief from the phantom limb pain. The quantitative and qualitative evaluation of the Mr.MAPP framework is performed by a group of able-bodied participants and Subject-Matter Experts (SME). The results indicate the effectiveness and usefulness of the Mr.MAPP framework. We also studied the effects of display technologies on the immersive experience and portability of the Mr.MAPP framework. The study indicates that the Oculus Rift provides a rich, immersive experience while it requires a high-end graphics enabled system. 3D TV provides a comfortable experience with partial immersion. Whereas, Samsung Gear VR provides an affordable, portable and moderate immersive solution. To incorporate the suggestions provided by Subject-Matter Experts, we plan to design a modified game tailored for lower limb amputees. Based on the encouraging preliminary findings about the usability and effectiveness of the Mr.MAPP framework, we are continuing our investigation with the help of rehab expert to evaluate the efficacy and feasibility of the proposed system for patients with limb amputees. As per SMEs, the Mr.MAPP framework has a great potential to improve traditional phantom limb pain management therapies.

3. Authentication for secure usage: To analyze the authenticity of 3D data captured using different types of short and long-range sensors, we carried out two different set of experiments:

For short-range sensors such as RGB-D cameras, we present a real-time 3D content manipulation framework to capture and manipulate live RGB-D data streams to create realistic images/videos showing individuals performing activities they did not actually do. The framework utilizes various computer vision and graphics techniques to render a photo-realistic animation of live 3D mesh models captured using RGB-D cameras. Subjective

analysis of manipulated RGB-D streams indicates that it was significantly difficult to distinguish between real and manipulated RGB-D streams. We also investigated the efficacy of noise analysis based forensic approach for identifying such manipulations.

For long-range sensors such as LiDAR, we have carried out an experimental study that opens up a new research area of forensic for LiDAR data. We have analyzed and identified possible attacks on the LiDAR data, which do not need additional hardware. Given the parameters for the modifications, these attacks can be carried out in real-time. We have also proposed two novel forensic approaches based on minimum inter-point distance and occlusion consistency for detecting additive forgery attacks. The analysis raises awareness to address possible threats to LiDAR data and to develop sophisticated forensic approaches for LiDAR data. Proposed attacks and forensic algorithm are also applicable to 3D point cloud data generated using other depth sensors as well.

Focusing on remote navigation in self-driving cars, we have proposed a novel framework, ALERT, for assisting autonomous vehicles and robots. ALERT has 4 key features:

- Handles the sparse nature and lack of surface information in LiDAR data with the proposed ordering scheme.
- Creates a real-time, dynamic watermark based on stereo RGB data and embeds it on 3D LiDAR data using fragile 3D watermarking schemes.
- Localizes the attacked region using cross-modal authentication and localization along with risk factor assessment.
- Bestows a possible, complete solution for a secure navigation of autonomous systems.

ALERT is evaluated in terms of imperceptibility and efficiency of watermarking technique, execution time performance and accuracy of both watermarking as well as a cross-modal module. The experimental results demonstrate the effectiveness of ALERT and its possible utilization in assisting autonomous vehicles and robots.

8.3 Future Work

1. Just Noticeable Difference based quality assessment: Based on the perceptual quality assessment performed in this work, we have identified the following directions for future work: a) Inability of existing perceptual quality metrics to adequately measure the visual quality of non-manifold meshes motivates the need to design a quality metric that correlates with the human perception. b) The future studies involving human responses based upon the concept of Just Noticeable Difference (JNDs) can be performed to assess and compare the subjective perceptual quality of various mesh simplification algorithms. Such analysis will assist in determining quality boundaries that will enable various adaptive streaming and rendering applications.

2. Extension of surface simplification for multiple RGB-D: The proposed curvature sensitive surface simplification method takes only a single RGB-D camera into account. While many mixed reality applications may involve capturing a live 3D avatar from multiple views to construct a watertight 3D model. Therefore it is important to extend the current surface simplification algorithm for incorporating multiple RGB-D cameras that can exploit the correlation between multiple views for efficient surface simplification.

3. Mr.MAPP patient study analysis: We expect to complete the ongoing patient study in the span of one year. The data gathered from this study can be utilized to understand the effectiveness of Mr.MAPP framework, as well as the patient's feedback, can be incorporated to improve the proposed system.

4. Extension of Mr.MAPP with visuo-tactile feedback: Perez-Marcos et al. (Perez-Marcos et al., 2012) demonstrate that it is possible to induce ownership of a virtual body with synchronous multisensory or sensorimotor correlations coherent with a fake body part. Such illusion may help in managing phantom pain effectively. Hence, a synchronous visuo-tactlie feedback can be incorporated in Mr.MAPP framework for enhanced illusion.

5. Generic forensic algorithm: Under the assumption of unavailability of color data for cross-modal verification or watermarking, it becomes crucial to design a generic forensic algorithm utilizing LiDAR data alone. The study on LiDAR forensics indicates that the proposed forensic algorithm can be effective only in case of additive types of attacks on 3D LiDAR data. Therefore, a more robust 3D forgery detection methods need to be developed that can address different types of attacks. Along with inconsistency in point density and occlusion effect, one can explore intrinsic properties of 3D LiDAR data such as perspective projection to build a generic forensic algorithm to validate LiDAR data.

6. Deep learning for 3D forensics: Instead of manual hand-crafted features, one can use deep learning techniques to adaptively learn the forgery detection from the observed situation. Mainly, deep learning approaches can be used in following ways:

- Deep learning based 3D object recognition and tracking can facilitate temporal forensic analysis.
- Generative adversarial networks (GANs) can be used to automatically generate forged 3D data which can be used for training a machine learning based forensic approach.
- Based on 3D object detection results, one can design a new watermarking scheme that will selectively embed the watermark in the 3D LiDAR data.

REFERENCES

Ackerman, E. (May 10, 2018). Drive.ai launches robot car pilot in texas with a focus on humans. `https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing`.

Adams, R. and L. Bischof (1994). Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence 16*(6), 641–647.

Agarwal, P. and B. Prabhakaran (2007). Robust blind watermarking mechanism for point sampled geometry. In *Proceedings of the 9th workshop on Multimedia & security*, pp. 175–186. ACM.

Agarwal, P. and B. Prabhakaran (2009). Robust blind watermarking of point-sampled geometry. *Information Forensics and Security, IEEE Transactions on 4*(1), 36–48.

Alexa, M., J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva (2003). Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on 9*(1), 3–15.

Arworks (2015). A flight into the new dimension of vr: When oculus meets kinect. `http://www.arworks.com/en/portfolio-item/oculus-and-kinect-in-one-system-a-new-dimension-of-vr`.

Aurenhammer, F. (1991). Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR) 23*(3), 345–405.

Awad, E., S. Dsouza, R. Kim, J. Schulz, J. Henrich, A. Shariff, J.-F. Bonnefon, and I. Rahwan (2018). The moral machine experiment. *Nature*, 1.

Azimi, M. (2012). Skeletal joint smoothing white paper. *MSDN digital library*.

B., C. (2014). 3d people surveillance on range data sequences of a rotating lidar. *Pattern Recognition Letters 50*, 149 – 158.

Bahirat, K., T. Annaswamy, and B. Prabhakaran (2017). Mr. mapp: Mixed reality for managing phantom pain. In *Proceedings of the 2017 ACM on Multimedia Conference*, pp. 1558–1566. ACM.

Bahirat, K. and B. Prabhakaran (2017). A study on lidar data forensics. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pp. 679–684. IEEE.

Barbosa, I., M. Cristani, A. Del Bue, L. Bazzani, and V. Murino (2012). Re-identification with rgb-d sensors. In *Computer Vision Ű ECCV 2012. Workshops and Demonstrations*, Volume 7583, pp. 433–442. Springer Berlin Heidelberg.

Besl, P. J. and R. Jain (1986). Invariant surface characteristics for 3d object recognition in range images. *Computer Vision, Graphics, and Image Processing 33*(1), 33–80.

Blakemore, S.-J., D. M. Wolpert, and C. D. Frith (2002). Abnormalities in the awareness of action. *Trends in cognitive sciences 6*(6), 237–242.

Boissonnat, J.-D. and F. Cazals (2001). Coarse-to-fine surface simplification with geometric guarantees. In *Computer Graphics Forum*, Volume 20, pp. 490–499. Wiley Online Library.

Boos, K., D. Chu, and E. Cuervo (2016). Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 291–304. ACM.

Bowman, D. A. and R. P. McMahan (2007). Virtual reality: how much immersion is enough? *Computer 40*(7).

Brodie, E. E., A. Whyte, and B. Waller (2003). Increased motor control of a phantom leg in humans results from the visual feedback of a virtual leg. *Neuroscience letters 341*(2), 167–169.

Brütsch, K., T. Schuler, A. Koenig, L. Zimmerli, S. Mérillat, L. Lünenburger, R. Riener, L. Jäncke, and A. Meyer-Heim (2010). Influence of virtual reality soccer game on walking performance in robotic assisted gait training for children. *Journal of neuroengineering and rehabilitation 7*(1), 15.

Cacciola, F. (2016). Triangulated surface mesh simplification. In *CGAL User and Reference Manual* (4.9 ed.). CGAL Editorial Board.

Canny, J. (1986, June). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*(6), 679–698.

Cayre, F. and B. Macq (2003). Data hiding on 3-d triangle meshes. *IEEE Transactions on signal Processing 51*(4), 939–949.

Chen, B. and G. W. Wornell (2001). Quantization index modulation methods for digital watermarking and information embedding of multimedia. *Journal of VLSI signal processing systems for signal, image and video technology 27*(1-2), 7–33.

Chen, S., H. Chung, and Y. Park (2013). Lidar scan and smart piezo layer combined damage detection.

Chen, X., A. Golovinskiy, and T. Funkhouser (2009). A benchmark for 3d mesh segmentation. In *Acm transactions on graphics (tog)*, Volume 28, pp. 73. ACM.

Chen, Y., M. Baran, H. Sundaram, and T. Rikakis (2011). A low cost, adaptive mixed reality system for home-based stroke rehabilitation. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 1827–1830. IEEE.

Chen, Y., N. Lehrer, H. Sundaram, and T. Rikakis (2010). Adaptive mixed reality stroke rehabilitation: system architecture and evaluation metrics. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, pp. 293–304. ACM.

Cheng, I. and P. Boulanger (2005). Feature extraction on 3-d texmesh using scale-space analysis and perceptual evaluation. *IEEE Transactions on Circuits and Systems for Video Technology 15*(10), 1234–1244.

Cheng, I. and P. Boulanger (2006, June). Adaptive online transmission of 3-d texmesh using scale-space and visual perception analysis. *IEEE Transactions on Multimedia 8*(3), 550–563.

Cheng, I., R. Shen, X.-D. Yang, and P. Boulanger (2006). Perceptual analysis of level-of-detail: The jnd approach. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pp. 533–540. IEEE.

Cho, W.-H., M.-E. Lee, H. Lim, and S.-Y. Park (2004). Watermarking technique for authentication of 3-d polygonal meshes. In *International Workshop on Digital Watermarking*, pp. 259–270. Springer.

Chou, C.-M. and D.-C. Tseng (2006). A public fragile watermarking scheme for 3d model authentication. *Computer-Aided Design 38*(11), 1154–1165.

Cignoni, P., M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia (2008). MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra (Eds.), *Eurographics Italian Chapter Conference*. The Eurographics Association.

Cignoni, P., C. Montani, and R. Scopigno (1998). A comparison of mesh simplification algorithms. *Computers & Graphics 22*.

Cignoni, P., C. Rocchini, and R. Scopigno (1996). Metro: Measuring error on simplified surfaces. Technical report, Paris, France, France.

Cignoni, P., C. Rocchini, and R. Scopigno (1998). Metro: Measuring error on simplified surfaces. In *Computer Graphics Forum*, Volume 17, pp. 167–174. Wiley Online Library.

Cohen, J., A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright (1996). Simplification envelopes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 119–128. ACM.

Cole, J. (2008). Virtual and augmented reality, phantom experience, and prosthetics. In *Psychoprosthetics*, pp. 141–153. Springer.

Cole, J., S. Crowle, G. Austwick, and D. Henderson Slater (2009). Exploratory findings with virtual reality for phantom limb pain; from stump motion to agency and analgesia. *Disability and rehabilitation 31*(10), 846–854.

Cotting, D., T. Weyrich, M. Pauly, and M. Gross (2004). Robust watermarking of point-sampled geometry. In *Shape Modeling Applications, 2004. Proceedings*, pp. 233–242. IEEE.

Demet, K., N. Martinet, F. Guillemin, J. Paysant, and J.-M. Andre (2003). Health related quality of life and related factors in 539 persons with amputation of upper and lower limb. *Disability and rehabilitation 25*(9), 480–486.

Desmond, D. M., K. O'neill, A. De Paor, G. McDarby, and M. MacLachlan (2006). Augmenting the reality of phantom limbs: three case studies using an augmented mirror box procedure. *JPO: Journal of Prosthetics and Orthotics 18*(3), 74–79.

Doerr, G. and J.-L. Dugelay (2003). A guide tour of video watermarking. *Signal processing: Image communication 18*(4), 263–282.

Domiter, V. and B. Zalik (2008). Sweep-line algorithm for constrained delaunay triangulation. *Int. J. Geogr. Inf. Sci. 22*(4), 449–462.

Duff, M., Y. Chen, S. Attygalle, J. Herman, H. Sundaram, G. Qian, J. He, and T. Rikakis (2010). An adaptive mixed reality training system for stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering 18*(5), 531–541.

Ebrahimi, T. (2009). Quality of multimedia experience: past, present and future. In *MM" 09: Proceedings of the seventeen ACM international conference on Multimedia*, Number MMSPL-CONF-2010-003, pp. 3–4. ACM.

Eck, M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle (1995). Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 173–182. ACM.

Egan, D., S. Brennan, J. Barrett, Y. Qiao, C. Timmerer, and N. Murray (2016). An evaluation of heart rate and electrodermal activity as an objective qoe evaluation method for immersive virtual reality environments. In *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*, pp. 1–6. IEEE.

Farid, H. (2009, March). Image forgery detection. *Signal Processing Magazine, IEEE 26*(2), 16–25.

Flor, H., T. Elbert, S. Knecht, C. Wienbruch, C. Pantev, et al. (1995). Phantom-limb pain as a perceptual correlate of cortical reorganization following arm amputation. *Nature 375*(6531), 482.

Francis, P. (September 3, 2006). At the scene of the crime. http://www.pobonline.com/articles/92344-at-the-scene-of-the-crime.

Fridrich, A. J., B. D. Soukal, and A. J. LukáŽ (2003). Detection of copy-move forgery in digital images. In *in Proceedings of Digital Forensic Research Workshop*.

Fridrich, J., D. Soukal, and J. Lukas (2003, August). Detection of copy move forgery in digital images. In *Digital Forensic Research Workshop*.

Gargantini, A., F. Terzi, M. Zambelli, and S. Bonfanti (2015). A low-cost virtual reality game for amblyopia rehabilitation. In *Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques*, pp. 81–84. ACM.

Garland, M. (1999). *Quadric-based polygonal surface simplification*. Ph. D. thesis, Georgia Institute of Technology.

Garland, M. and P. S. Heckbert (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 209–216.

Garland, M. and P. S. Heckbert (1998). Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the conference on Visualization'98*, pp. 263–269. IEEE Computer Society Press.

Gautier, J., O. Le Meur, and C. Guillemot (2012). Efficient depth map compression based on lossless edge coding and diffusion. In *Picture Coding Symposium (PCS), 2012*, pp. 81–84. IEEE.

Geiger, A., P. Lenz, C. Stiller, and R. Urtasun (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

Gent, E. (2017). Should self-driving cars make ethical decisions like we do? https://singularityhub.com/2017/07/11/should-self-driving-cars-make-ethical-decisions-like-we-do/#sm.00014wpdjepxee2vqmr24m3twq764.

Goldman, R. (2005). Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design 22*(7), 632–658.

Guéziec, A., G. Taubin, F. Lazarus, and W. Horn (1998). Converting sets of polygons to manifold surfaces by cutting and stitching. In *Visualization'98. Proceedings*, pp. 383–390. IEEE.

Guizzo, E. (2011, October). How Google's Self-Driving Car Works. Online.

Hachicha, W., A. Beghdadi, and F. A. Cheikh (2013). Stereo image quality assessment using a binocular just noticeable difference model. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pp. 113–117. IEEE.

He, T., L. Hong, A. Kaufman, A. Varshney, and S. Wang (1995). Voxel based object simplification. In *Proceedings of the 6th conference on Visualization'95*, pp. 296. IEEE Computer Society.

Heckbert, P. S. and M. Garland (1999). Optimal triangulation and quadric-based surface simplification. *Computational Geometry 14*(1âĂŞ3), 49 – 65.

Heger, F. W. and S. Singh (2006). Sliding autonomy for complex coordinated multi-robot tasks: Analysis & experiments.

Hoppe, H. (1996). Progressive meshes. In *23rd annual conference on Computer graphics and interactive techniques*, pp. 99–108. ACM.

Hoppe, H. (1999). New quadric metric for simplifiying meshes with appearance attributes. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pp. 59–66. IEEE Computer Society Press.

Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle (1993). Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 19–26. ACM.

Hosseini, M., D. T. Ahmed, and S. Shirmohammadi (2012). Adaptive 3d texture streaming in m3g-based mobile games. In *Proceedings of the 3rd Multimedia Systems Conference*, pp. 143–148. ACM.

Hou, W., Z. Xu, N. Qin, D. Xiong, and M. Ding (2015). Surface reconstruction through poisson disk sampling. *PLoS ONE 10*, 1–15.

Houshiar, H., J. Elseberg, D. Borrmann, and A. Nüchter (2015). A study of projections for key point based registration of panoramic terrestrial 3d laser scan. *Geo-spatial Information Science 18*(1), 11–31.

Huang, C.-H., E. Boyer, and S. Ilic (2013). Robust human body shape and pose tracking. In *3DTV-Conference, 2013 International Conference on*, pp. 287–294. IEEE.

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.

Jing, L., W. Yinghui, H. Wenjuan, and L. Ye (2014). A new watermarking method of 3d mesh model. *TELKOMNIKA Indonesian Journal of Electrical Engineering 12*(2), 1610–1617.

Jones, T. (February 2, 2011). Scanning for forensic criminal defense presentations. `http://www.landairsurveying.com/blog/scanning-for-forensic-presentations/`.

Kashani, A. G., P. S. Crawford, S. K. Biswas, A. J. Graettinger, and D. Grau (2014). Automated tornado damage assessment and wind speed estimation based on terrestrial laser scanning. *Journal of Computing in Civil Engineering 29*(3), 04014051.

Katz, J. (1992). Psychophysiological contributions to phantom limbs.

Keighrey, C., R. Flynn, S. Murray, and N. Murray (2017). A qoe evaluation of immersive augmented and virtual reality speech & language assessment applications. In *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*, pp. 1–6. IEEE.

Kim, S.-Y. and Y.-S. Ho (2007). Mesh-based depth coding for 3d video using hierarchical decomposition of depth maps. In *2007 IEEE International Conference on Image Processing*, Volume 5, pp. V–117. IEEE.

Kreylos, O. (2005). Collaborative exploration of medical datasets using virtual environments. `http://graphics.cs.ucdavis.edu/~staadt/research_collaborative_vis.html`.

Ladicky, L., P. H. Torr, and A. Zisserman (2013). Human pose estimation using a joint pixel-wise and part-wise formulation. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3578–3585.

Lavoué, G. (2011). A multiscale metric for 3d mesh visual quality assessment. In *Computer Graphics Forum*, Volume 30, pp. 1427–1437. Wiley Online Library.

Lavoué, G., E. D. Gelasca, F. Dupont, A. Baskurt, and T. Ebrahimi (2006). Perceptually driven 3d distance metrics with application to watermarking. In *Applications of Digital Image Processing XXIX*, Volume 6312, pp. 63120L. International Society for Optics and Photonics.

Lecuyer, A., S. Coquillart, A. Kheddar, P. Richard, and P. Coiffet (2000). Pseudo-haptic feedback: Can isometric input devices simulate force feedback? In *Virtual Reality, 2000. Proceedings. IEEE*, pp. 83–90. IEEE.

Levinson, J., J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun (2011, June). Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 163–168.

Levoy, M., J. Gerth, B. Curless, and K. Pull (2005). The stanford 3d scanning repository. *URL http://www-graphics. stanford. edu/data/3dscanrep*.

Lin, H.-Y., H.-Y. Liao, C.-S. Lu, and J.-C. Lin (2005). Fragile watermarking for authenticating 3-d polygonal meshes. *IEEE Transactions on Multimedia 7*(6), 997–1006.

Lindstrom, P. and G. Turk (1998). Fast and memory efficient polygonal simplification. In *Visualization'98. Proceedings*, pp. 279–286.

Linsen, L. (2001). *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek.

Liu, Y. and J. Zhao (2010). A new video watermarking algorithm based on 1d dft and radon transform. *Signal Processing 90*(2), 626–639.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, Volume 21, pp. 163–169. ACM.

Lounsbery, M., T. D. DeRose, and J. Warren (1997). Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics (TOG) 16*(1), 34–73.

Luebke, D. and C. Erikson (1997). View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 199–208. ACM Press/Addison-Wesley Publishing Co.

Luebke, D. P. (2001). A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications 21*.

MacLachlan, M., D. McDonald, and J. Waloch (2004). Mirror treatment of lower limb phantom pain: a case study. *Disability and rehabilitation 26*(14-15), 901–904.

McFarland, M. (2016). Tesla driver killed in autopilot crash said the technology was 'great'. `http://money.cnn.com/2016/07/01/technology/tesla-driver-death-autopilot/index.html?iid=EL`.

McFarland, M. (2018). Uber self-driving car kills pedestrian in first fatal autonomous crash. `http://money.cnn.com/2018/03/19/technology/uber-autonomous-car-fatal-crash/index.html?iid=EL`.

McMahan, R. P., D. A. Bowman, D. J. Zielinski, and R. B. Brady (2012). Evaluating display fidelity and interaction fidelity in a virtual reality game. *IEEE Transactions on Visualization & Computer Graphics* (4), 626–633.

Medimegh, N., S. Belaid, and N. Werghi (2015). A survey of the 3d triangular mesh watermarking techniques. *Int J Multimed 1*(1).

Meehan, M., B. Insko, M. Whitton, and F. P. Brooks Jr (2002). Physiological measures of presence in stressful virtual environments. *Acm transactions on graphics (tog) 21*(3), 645–652.

Mehrotra, S., Z. Zhang, Q. Cai, C. Zhang, and P. A. Chou (2011). Low-complexity, near-lossless coding of depth maps from kinect-like depth cameras. In *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, pp. 1–6. IEEE.

Melzack, R. (1975). The mcgill pain questionnaire: major properties and scoring methods. *Pain 1*(3), 277–299.

Metry, M. (2017). Vr is revolutionizing trauma training at children's hospital los angeles. `http://www.vudream.com/vr-revolutionizing-trauma-training-childrens-hospital-los-angeles/`.

Microsoft (2015). Performance recommendations for microsoft hololens. `https://developer.microsoft.com/en-us/windows/holographic/performance_recommendations`.

Microsoft (2016). Kinect-enabled vr holds productivity promise. `https://blogs.msdn.microsoft.com/kinectforwindows/2016/06/09/kinect-enabled-vr-holds-productivity-promise`.

Milani, S., M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro (2012). An overview on video forensics. *APSIPA Transactions on Signal and Information Processing 1*.

Moenning, C. and N. A. Dodgson (2003). A new point cloud simplification algorithm. In *Proc. Int. Conf. on Visualization, Imaging and Image Processing*, pp. 1027–1033.

Murray, C. D., S. Pettifer, T. Howard, E. Patchick, F. Caillette, and J. Murray (2009). Virtual solutions to phantom problems: Using immersive virtual reality to treat phantom limb pain. In *Amputation, Prosthesis Use, and Phantom Limb Pain*, pp. 175–196. Springer.

Nüchter, A. and K. Lingemann (2010). Robotic 3d scan repository. `http://kos.informatik.uni-osnabrueck.de/3Dscans`.

Oh, K.-J., A. Vetro, and Y.-S. Ho (2011). Depth coding using a boundary reconstruction filter for 3-d video systems. *IEEE Transactions on Circuits and Systems for Video Technology 21*(3), 350–359.

Ohbuchi, R., A. Mukaiyama, and S. Takahashi (2004). Watermarking a 3d shape model defined as a point set. In *Cyberworlds, 2004 International Conference on*, pp. 392–399. IEEE.

Olofsson, A. (2010). Modern stereo correspondence algorithms: investigation and evaluation.

O'Neill, K., A. de Paor, M. MacLachlan, and G. McDarby (2003). An investigation into the performance of a virtual mirror box for the treatment of phantom limb pain in amputees using augmented reality technology. In *Human-Computer-Interaction International*. Lawrence Erlbaum Associates Inc.

Ortiz-Catalan, M., R. A. Guðmundsdóttir, M. B. Kristoffersen, A. Zepeda-Echavarria, K. Caine-Winterberger, K. Kulbacka-Ortiz, C. Widehammar, K. Eriksson, A. Stockselius, C. Ragnö, et al. (2016). Phantom motor execution facilitated by machine learning and augmented reality as treatment for phantom limb pain: a single group, clinical trial in patients with chronic intractable phantom limb pain. *The Lancet 388*(10062), 2885–2894.

Ortiz-Catalan, M., N. Sander, M. B. Kristoffersen, B. Håkansson, and R. Brånemark (2014). Treatment of phantom limb pain (plp) based on augmented reality and gaming controlled by myoelectric pattern recognition: a case study of a chronic plp patient.

Ovreiu, E. (2012). *Accurate 3D mesh simplification*. Ph. D. thesis, INSA de Lyon.

Pajarola, R., M. Sainz, and Y. Meng (2003). *Depth-mesh objects: Fast depth-image meshing and warping*. Information and Computer Science, University of California, Irvine.

Pauly, M., M. Gross, and L. P. Kobbelt (2002). Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pp. 163–170. IEEE Computer Society.

Perez-Marcos, D., M. Solazzi, W. Steptoe, W. Oyekoya, A. Frisoli, T. Weyrich, A. Steed, F. Tecchia, M. Slater, and M. V. Sanchez-Vives (2012). A fully immersive set-up for remote interaction and neurorehabilitation based on virtual body ownership. *Frontiers in neurology 3*, 110.

Petit, J. and S. Shladover (2015). Potential cyberattacks on automated vehicles. *Intelligent Transportation Systems, IEEE Transactions on 16*, 546–556.

Petit, J., B. Stottelaar, M. Feiri, and F. Kargl (2015, 11/2015). Remote attacks on automated vehicles sensors: Experiments on camera and lidar. In *Black Hat Europe*.

Pohl, D. and C. F. de Tejada Quemada (2016). See what i see: Concepts to improve the social acceptance of hmds. In *2016 IEEE Virtual Reality (VR)*, pp. 267–268. IEEE.

Pojar, E. and D. Schmalstieg (2003). User-controlled creation of multiresolution meshes. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pp. 127–130. ACM.

Popescu, A. C. and H. Farid (2004). Exposing digital forgeries by detecting duplicated image regions. Technical report.

Praun, E., H. Hoppe, and A. Finkelstein (1999). Robust mesh watermarking. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 49–56. ACM Press/Addison-Wesley Publishing Co.

Pruett, C. (2015). Squeezing performance out of your unity gear vr game. `https://developer.oculus.com/blog/squeezing-performance-out-of-your-unity-gear-vr-game`.

Puig, J., A. Perkis, F. Lindseth, and T. Ebrahimi (2012). Towards an efficient methodology for evaluation of quality of experience in augmented reality. In *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*, pp. 188–193. IEEE.

Qi, K., X. Dong-qing, and Z. Da-fang. A robust watermarking scheme for 3d point cloud models using self-similarity partition. In *Wireless Communications, Networking and Information Security 2010*, pp. 287–291.

Qin, S., S. Ge, H. Yin, L. Liu, and I. Heynderickx (2009). Effect of experimental methodology on the jnd of the black level for natural images. *Journal of the Society for Information Display 17*(8), 687–694.

Raake, A. and S. Egger (2014). Quality and quality of experience. In *Quality of experience*, pp. 11–33. Springer.

Raghuraman, S., K. Bahirat, and B. Prabhakaran (2015a). Evaluating the efficacy of rgb-d cameras for surveillance. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pp. 1–6. IEEE.

Raghuraman, S., K. Bahirat, and B. Prabhakaran (2015b). Evaluating the efficacy of rgb-d cameras for surveillance. In *ICME*, pp. 1–6. IEEE.

Raghuraman, S., K. Venkatraman, Z. Wang, B. Prabhakaran, and X. Guo (2013a). A 3d tele-immersion streaming approach using skeleton-based prediction. In *ACM Multimedia*, pp. 721–724.

Raghuraman, S., K. Venkatraman, Z. Wang, B. Prabhakaran, and X. Guo (2013b). A 3d tele-immersion streaming approach using skeleton-based prediction. In *Proceedings of the 21st ACM international conference on Multimedia*, pp. 721–724. ACM.

Ramachandran, V. S. and D. Rogers-Ramachandran (1996). Synaesthesia in phantom limbs induced with mirrors. *Proceedings of the Royal Society of London B: Biological Sciences 263*(1369), 377–386.

Romanoni, A., A. Delaunoy, M. Pollefeys, and M. Matteucci (2016). Automatic 3d reconstruction of manifold meshes via delaunay triangulation and mesh sweeping. In *Winter Applications of Computer Vision*.

Rossignac, J. and P. Borrel (1993). *Multi-resolution 3D approximations for rendering complex scenes*. Springer.

Sauer, M. (2011). Mixed-reality for enhanced robot teleoperation.

Schroeder, W. J., J. A. Zarge, and W. E. Lorensen (1992). Decimation of triangle meshes. In *ACM Siggraph Computer Graphics*, Volume 26, pp. 65–70. ACM.

Sherman, R. A., C. J. Sherman, and L. Parker (1984). Chronic phantom and stump pain among american veterans: results of a survey. *Pain 18*(1), 83–95.

Shotton, J., T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM 56*(1), 116–124.

Shuai, L., C. Li, X. Guo, B. Prabhakaran, and J. Chai (2017). Motion capture with ellipsoidal skeleton using multiple depth cameras. *IEEE transactions on visualization and computer graphics 23*(2), 1085–1098.

Siddiqui, M. and G. Medioni (2010). Human pose estimation from a single view point, real-time range sensor. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 1–8. IEEE.

Simpson, R. (1994). Anisotropic mesh transformations and optimal error control. *Applied Numerical Mathematics 14*, 183 – 198.

State of California, D. (February 26, 2018a). Testing of autonomous vehicles. `https://www.dmv.ca.gov/portal/wcm/connect/a6ea01e0-072f-4f93-aa6c-e12b844443cc/DriverlessAV_Adopted_Regulatory_Text.pdf?MOD=AJPERES`.

State of California, D. (February 26, 2018b). Testing of autonomous vehicles with a driver. `https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing`.

Steed, A., S. Frlston, M. M. Lopez, J. Drummond, Y. Pan, and D. Swapp (2016). An 'in the wild' experiment on presence and embodiment using consumer virtual reality equipment. *IEEE transactions on visualization and computer graphics 22*(4), 1406–1414.

Stiller, C., J. Hipp, C. Rössig, and A. Ewald (2000). Multisensor obstacle detection and tracking. *Image and vision Computing 18*(5), 389–396.

Stiller, C. and J. Ziegler (2012, March). 3d perception and planning for self-driving and cooperative automobiles. In *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, pp. 1–7.

Tech, G., K. Wegner, Y. Chen, and S. Yea (2012). 3d-hevc test model 2. *ITU-T SG 16*.

Timmerer, C., T. Ebrahimi, and F. Pereira (2015). Toward a new assessment of quality. *IEEE Computer 48*(3), 108–110.

Tong, X., D. Gromala, A. Amin, and A. Choo (2015). The design of an immersive mobile virtual reality serious game in cardboard head-mounted display for pain management. In *International Symposium on Pervasive Computing Paradigms for Mental Health*, pp. 284–293. Springer.

Torkhani, F., K. Wang, and J.-M. Chassery (2014). A curvature-tensor-based perceptual quality metric for 3d triangular meshes. *Machine Graphics and Vision 23*(1-2), 59–82.

Turk, G. (1992). Re-tiling polygonal surfaces. *ACM SIGGRAPH Computer Graphics 26*(2), 55–64.

Unity3D (2012). Vertex limit for unity3d. `http://answers.unity3d.com/questions/255405/vertex-limit.html`.

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.

Vieira, J., M. Sousa, A. Arsénio, and J. Jorge (2015). Augmented reality for rehabilitation using multimodal feedback. In *Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques*, pp. 38–41. ACM.

W., C.-Y., H.-H. C., C.-K. L., and W.-C. Y. (2013). A study of applying light detection and ranging (lidar) to crime scene documentation. *Forensic Science Journal 12*, 31–46.

Wang, H., I. Katsavounidis, J. Zhou, J. Park, S. Lei, X. Zhou, M.-O. Pun, X. Jin, R. Wang, X. Wang, et al. (2017). Videoset: A large-scale compressed video quality dataset based on jnd measurement. *Journal of Visual Communication and Image Representation 46*, 292–302.

Wang, J., J. Feng, and Y. Miao (2012). A robust confirmable watermarking algorithm for 3d mesh based on manifold harmonics analysis. *The Visual Computer 28*(11), 1049–1062.

Wang, K. (2012). Source code for fast mesh perceptual distance (fmpd). `http://www.gipsa-lab.grenoble-inp.fr/~kai.wang/publications_en.html`.

Wang, K., G. Lavoué, F. Denis, and A. Baskurt (2008). A comprehensive survey on three-dimensional mesh watermarking. *IEEE Transactions on Multimedia 10*(8), 1513–1527.

Wang, K., F. Torkhani, and A. Montanvert (2012). A fast roughness-based approach to the assessment of 3d mesh visual quality. *Computers & Graphics 36*(7), 808–818.

Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*(4), 600–612.

Wasenmüller, O., M. Meyer, and D. Stricker (2016, March). CoRBS: Comprehensive rgb-d benchmark for slam using kinect v2. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Volume . IEEE.

Watson, A. B. and A. Fitzhugh (1990). The method of constant stimuli is inefficient. *Perception & psychophysics 47*(1), 87–91.

Wu, J.-H., S.-M. Hu, C.-L. Tai, and J.-G. Sun (2001). An effective feature-preserving mesh simplification scheme based on face constriction. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pp. 12–21. IEEE.

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, pp. 13–22.

Yang, X., W. Ling, Z. Lu, E. P. Ong, and S. Yao (2005). Just noticeable distortion model and its applications in video coding. *Signal Processing: Image Communication 20*(7), 662–680.

Yeo, B. and M. Yeung (1999a). Watermarking 3d objects for verification. *Computer Graphics and Applications, IEEE 19*(1), 36–45.

Yeo, B.-L. and M. M. Yeung (1999b). Watermarking 3d objects for verification. *IEEE Computer Graphics and Applications 19*(1), 36–45.

Yin, K., Z. Pan, J. Shi, and D. Zhang (2001). Robust mesh watermarking based on multiresolution processing. *Computers & graphics 25*(3), 409–420.

Yu, Z., H. H. Ip, and L. Kwok (2003). A robust watermarking scheme for 3d triangular mesh models. *Pattern recognition 36*(11), 2603–2614.

Zafeiriou, S., A. Tefas, and I. Pitas (2005). Blind robust watermarking schemes for copyright protection of 3d mesh objects. *IEEE Transactions on Visualization and Computer Graphics 11*(5), 596–607.

Zhang, X., W. Lin, and P. Xue (2008). Just-noticeable difference estimation with pixels in images. *Journal of Visual Communication and Image Representation 19*(1), 30–41.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia 19*(2), 4–10.

Zhao, Y., Z. Chen, C. Zhu, Y.-P. Tan, and L. Yu (2011). Binocular just-noticeable-difference model for stereoscopic images. *IEEE Signal Processing Letters 18*(1), 19–22.

Zheng, B., Y. Zhao, J. C. Yu, K. Ikeuchi, and S. Zhu (2013, June). Beyond point clouds: Scene understanding by reasoning geometry and physics. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Ziegler-Graham, K., E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer (2008). Estimating the prevalence of limb loss in the united states: 2005 to 2050. *Archives of physical medicine and rehabilitation 89*(3), 422–429.

Zielinski, D. J., M. A. Sommer, H. M. Rao, L. G. Appelbaum, N. D. Potter, and R. Kopper (2016). Evaluating the effects of image persistence on dynamic target acquisition in low frame rate virtual environments. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 133–140. IEEE.

BIOGRAPHICAL SKETCH

Kanchan Bahirat was brought up in Mumbai (India), one of the busiest city in the world. Since childhood, Kanchan had a deep interest in mathematics and science. It seemed only natural for her to take up undergraduate studies in engineering. She graduated from Mumbai University with the First class and Distinction for her Bachelor of Engineering in Electronics and Telecommunication in 2007. Next, she continued her quest for learning by earning her Master of Technology in Electrical Engineering from the Indian Institute of Technology Bombay. The research work done during her master's sparked a deep interest and motivation to explore the area of computer vision and graphics. After completing her master's, Kanchan joined a Mumbai based company, Vubites India Pvt. Ltd. (a subsidiary of Rediff.com) as a product engineer where she worked on various computer vision and image processing techniques for facilitating an automatic advertisement insertion in live television broadcast. Although she was excelling at her work, her desire to explore and dive deep into computer vision research lead her to continue her research as a PhD student at The University of Texas at Dallas. During her PhD, she designed and explored various computer vision and graphics techniques with special focus on 3D Tele-immersion, Virtual and Augmented reality for healthcare domain applications. Her work has opened up a new research area of 3D forensics which has a significant impact on various sensitive applications such as self-driving cars and 3D surveillance.

CURRICULUM VITAE

Kanchan Bahirat

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: Kanchan.Bahirat@utdallas.edu

Educational History:

B.E., Electronics and Telecommunication, Mumbai University, India, 2007
M.Tech., Electrical Engineering, Indian Institute of Technology Bombay, 2011
Ph.D., Computer Science, The University of Texas at Dallas, 2018

Domain Adapted Semi-Supervised Classification of Remote Sensing Images
Masters Thesis
Department of Electrical Engineering, Indian Institute of Technology Bombay
Advisor: Dr. Subhasis Chaudhuri

On 3D Content Manipulation: Simplification, Modification and Authentication
Ph.D. Dissertation
Department of Computer Science, The University of Texas at Dallas
Advisor: Dr. Balakrishnan Prabhakaran

Employment History:

Research and Teaching Assistant, The University of Texas at Dallas, August 2013 – December 2018
Research Intern, Sony US Research Center, June 2017 – August 2017
Product Engineer, Vubites India Pvt. Limited, July 2011 – July 2013

Professional Recognition and Honors:

ACM Multimedia 2017 nominated for Best Paper Award.
Received "The L'oréal India for Young Women in Science Scholarship" in 2003, sponsoring the entire undergraduate engineering education.
Conferences: ACM MM 2018, ACM MM 2017, ACM MMSys 2017, IEEE ICME 2017, IEEE ICME 2015, IEEE ISM 2016, ACM MMSys 2016.
Reviewer for: IEEE Transaction on Multimedia, IEEE VR, Multimedia Systems Journal.

Publications:

1. **K. Bahirat**, U. Shah, A. Cardenas, B. Prabhakaran, "ALERT: Adding a Secure Layer in Decision Support for Advanced Driver Assistance System (ADAS)". In Proceedings of the 2018 ACM on Multimedia Conference (In press).

2. **K.Bahirat**, C. Lai, RP. McMahan, B. Prabhakaran, "Designing and Evaluating A Mesh Simplification Algorithm for Virtual Reality". ACM Transactions on Multimedia Computing Communications and Applications (TOMM) 14.3s (2018): 63.

3. **K. Bahirat**, S. Raghuraman, B. Prabhakaran, "Real-time, Curvature-Sensitive Surface Simplification Using Depth Images". IEEE Transaction on Multimedia 2017.

4. **K. Bahirat**, T. Annaswamy, B. Prabhakaran, "Mr.MAPP: Mixed Reality for MAnaging Phantom Pain". In Proceedings of the 2017 ACM on Multimedia Conference, pp. 1558-1566. ACM, 2017 (nominated as *a best paper candidate* from a novel papers track).

5. **K. Bahirat**, C. Lai, RP. McMahan, B. Prabhakaran, "A Boundary and Texture Preserving Mesh Simplification Algorithm for Virtual Reality". In Proceedings of the 8th ACM on Multimedia Systems Conference, pp. 50-61. ACM, 2017.

6. **K. Bahirat**, B. Prabhakaran, "A study on lidar data forensics". In Multimedia and Expo (ICME), 2017 IEEE International Conference on, pp. 679-684. IEEE, 2017.

7. S. Raghuraman, **K. Bahirat**, B. Prabhakaran, "A Visual Latency Estimator for 3D Tele-Immersion". In Proceedings of the 8th ACM on Multimedia Systems Conference, pp. 272-283. ACM, 2017.

8. K. Desai, **K. Bahirat**, B. Prabhakaran, "Learning-based objective evaluation of 3D human open meshes". In 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 733-738. IEEE, 2017.

9. K. Desai, **K. Bahirat**, S. Ramalingam, B. Prabhakaran, T. Annaswamy and U. Makris, "Augmented Reality-based Exergames for Rehabilitation". In Proceedings of the 7th International Conference on Multimedia Systems, p. 22. ACM, 2016.

10. K. Desai, **K. Bahirat**, S. Raghuraman and B. Prabhakaran, "Network Adaptive Textured Mesh Generation for Collaborative 3D Tele-Immersion". In Multimedia (ISM), 2015 IEEE International Symposium on, pp. 107-112. IEEE, 2015.

11. S. Raghuraman, **K. Bahirat**, and B. Prabhakaran, "Evaluating the efficacy of RGB-D cameras for surveillance". In 2015 IEEE International Conference on Multimedia and Expo (ICME), pp. 1-6. IEEE, 2015.