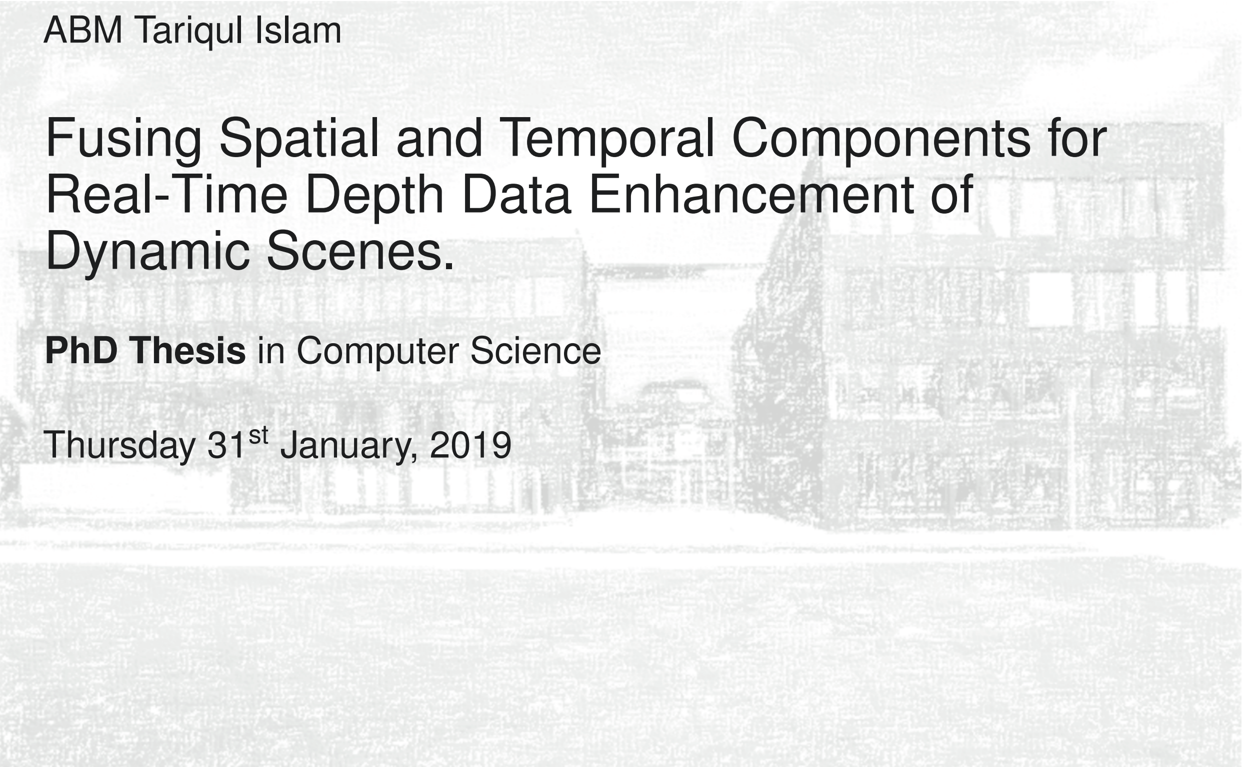
Universität Rostock

Traditio et Innovatio

Chair of Visual Computing

Institute for Visual and Analytic Computing



ABM Tariqul Islam

Fusing Spatial and Temporal Components for Real-Time Depth Data Enhancement of Dynamic Scenes.

PhD Thesis in Computer Science

Thursday 31st January, 2019

Please cite as:
ABM Tariqul Islam, “Fusing Spatial and Temporal Components for Real-Time Depth Data Enhancement of Dynamic Scenes.,” PhD Thesis (Doctorarbeit), University of Rostock, Institute for Visual and Analytic Computing, October 2019.

University of Rostock
Faculty of Computer Science and Electrical Engineering
Institute for Visual and Analytic Computing

Albert-Einstein-Str. 22 · 18059 Rostock · Germany

https://www.informatik.uni-rostock.de/

Fusing Spatial and Temporal Components for Real-Time Depth Data Enhancement of Dynamic Scenes.

PhD Thesis
to obtain the academic degree of
Doktor-Ingenieur (Dr.-Ing.)
of the Faculty of Computer Science and Electrical Engineering
at the University of Rostock, Germany

Submitted by
ABM Tariqul Islam
born on 02.01.1982 in Khulna, Bangladesh

Study and research carried out at
Visual Computing Group, Institute for Visual and Analytic Computing
University of Rostock, Germany

Rostock, 2019

gegründet 1419

https://doi.org/10.18453/rosdok_id00002542

First Reviewer (Supervisor): **Prof. Dr. sc. techn. Oliver Staadt**, Visual Computing Group, Institute for Visual and Analytic Computing, University of Rostock, Germany

Second Reviewer: **Prof. Dr. Renato Pajarola**, Visualization and Multimedia Lab, Department of Informatics, University of Zurich, Switzerland

Year of submission: **2019**

Year of defense: **2019**

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.

I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.

(ABM Tariqul Islam)
Rostock, Thursday 31st January, 2019

Abstract

Recent advances in camera technologies have made depth-sensing devices, such as the LIDAR scanners, light field cameras, structured light devices and Time-of-Flight (ToF) devices available. While high-end devices such as LIDAR scanners and light field cameras provide comparatively interference-free depth images with high accuracy, devices based on structured light or ToF show lower resolution and artifacts such as lack of depth, holes, flickering, inhomogeneity and alike. On the other hand, high-end devices are quite expensive and often difficult to operate due to their size and weight. Consequently, both the research and industry are more likely to turn to the readily available and less expensive devices such as the structured light devices and ToF devices. Although these devices capture a scene with reasonable resolution and speed, the depth data exhibits a substantial amount of artifacts. The artifacts generated by these consumer depth cameras come in different forms. One common artifact appears as randomly distributed holes over the surface of the objects of a scene; especially, where the depth discontinuity occurs. Often, such artifact is spread over the temporal domain and causes the flickering artifacts, meaning the holes appear and disappear at random locations on the object's surfaces over the successive frames. Moreover, in case of dramatic or drastic movement of the objects in a scene, ghosting artifacts are often perceived on the depth frames when post-processing is applied on these depth frames. Hence, the depth images from consumer depth cameras usually need further enhancement otherwise, they cannot be used in various crucial real-world applications. For example, in object tracking applications, the noise on an object's surface greatly deteriorates the tracking performance and in forensic analysis of a crime scene, where every detail is vital, a poorly reconstructed scene might hamper the recovery of correct information. Likewise, in telepresence or e-learning systems, low-quality 3D scene limits the sensation of a natural presence of the remote users to the local site.

The primary focus of this work is on the qualitative improvement of depth data recorded with the low-cost depth cameras. In addition to noise reduction, this primarily concerns the reduction of the described artifacts. In this context, this thesis proposes a new concept for real-time calculation of high-quality depth images. The main contribution is the development of a new depth image enhancement filter that fuses the spatial and temporal information of depth images in real time and thus, stabilizes and enhances the distorted depth data. Furthermore, this thesis presents a noise visualization and analysis method with the aim to suppress the inherent noise of the depth values and, eventually, to optimize the proposed depth enhancement method. In order to better understand the characteristics of depth noise and ultimately remove the noise, the analysis method is applied to ground truth test data that are generated from experiments using precise tracking information. For the evaluation of the proposed real-time depth enhancement strategy, experimental results are compared with other state-of-the-art methods on reference data sets. The results show that noise and the number of flickering holes are significantly minimized and ghosting artifacts are successfully removed.

Furthermore, this thesis presents two strategies for real-time camera data reduction, with which the processed images can be transmitted without noticeable delay. This is especially true for the multi-camera configurations used in many applications, which deliver image streams from multiple cameras simultaneously. The developed data reduction methods function as pre-processing steps for the transmission of scenes that are recorded from several angles and therefore contain large amounts of image data about the recording location. Extensive testing shows that the reduction strategies successfully reduce the amount of transmission data and hence, enable uninterrupted transmission in a low-bandwidth network.

Zusammenfassung

Mit den kürzlich erfolgten Fortschritten in der Kameratechnologie wurden Tiefenmessgeräte wie der LIDAR-Scanner, Lichtfeldkameras, Vorrichtungen, die mit strukturiertem Licht arbeiten, und sogenannte Time-of-Flight (ToF) Geräte verfügbar. Während High-End-Geräte wie die LIDAR- Scanner und Lichtfeldkameras vergleichsweise störungsfreie Tiefenbilder mit hoher Genauigkeit liefern, zeigen Geräte, die mit strukturiertem Licht oder auf Basis von ToF funktionieren, eine niedrigere Auflösung und Artefakte wie fehlende Tiefe, Löcher, Flackern oder Inhomogenität. Andererseits sind High-End-Geräte aufgrund ihrer Größe und ihres Gewichts recht teuer und oft schwer zu bedienen. Folglich wenden sich sowohl Forschung als auch Industrie häufiger den leicht verfügbaren und kostengünstigeren Geräten zu. Neben heftigem Rauschen, stellt die beträchtliche Menge an Artefakten, die in verschiedenen Formen auftreten, Hauptproblem bei der Arbeit mit diesen Geräten dar. Besonders häufig erscheinen zufällig verteilte Löcher in der Tiefeninformation von Oberflächen der Szenenobjekte. Insbesondere geschieht dies an abrupten Änderungen der Tiefenwerte, d.h. in der Nähe von Objektkanten. Zudem tritt dieses Artefakt häufig diskontinuierlich über die Zeitdomäne verteilt auf und verursacht so ein stark auffälliges Flackern, bei dem die Löcher in aufeinanderfolgenden Frames immer wieder erscheinen und verschwinden. Weniger zufällig als Löcher entstehen durch die notwendige Kompensation schneller Bewegungen von Szenenobjekten oft unerwünschte Geisterbilder in den Tiefendaten. Zur Verwendbarkeit für praktische Anwendungen müssen Tiefenbilder deshalb in der Regel intensiv weiterverarbeitet werden, um sie von diesen Artefakten zu befreien. Ansonsten ist die rekonstruierte 3D-Szene von minderer Güte und nicht adäquat für die jeweilige Anwendung. Beispielsweise vermindert sich in Telepräsenz- oder E-Learning- Systemen mit der Qualität der Szenenrekonstruktion das Gefühl der natürlichen Präsenz von entfernten Benutzer am lokalen Standort und damit gleichermaßen die Immersion. Auf ähnliche Weise kann eine fehlerhafte 3D-Nachbildung eines Tatorts eine forensische Analyse erschweren, weil falsch dargestellte Details den entscheidenden Hinweis verschleiern können.

Der primäre Fokus der vorliegenden Arbeit liegt auf der qualitativen Verbesserung von Tiefeninformationen, die mit preiswerten Tiefenkameras aufgenommen werden. Dies betrifft neben Rauschunterdrückung in erster Linie die Reduzierung der beschriebenen

Artefakte. In diesem Zusammenhang schlägt die Arbeit ein neues Konzept zur Echtzeit-Berechnung qualitativ hochwertiger Tiefenbilder vor. Der Hauptbeitrag dabei ist die Entwicklung eines neuen Tiefendatenfilters, der die räumlichen und zeitlichen Informationen aus Tiefenbildern in Echtzeit kombiniert und sie damit stabilisiert und verbessert. Weiterhin präsentiert die Arbeit ein Rauschvisualisierungs- und Analyseverfahren mit dem Ziel, das inhärente Rauschen der Tiefenwerte unterdrücken zu können. Um die Charakteristika des Tiefenrauschens besser zu verstehen und um das Rauschen letztendlich entfernen zu können, wird das Analyseverfahren auf Ground-Truth-Testdaten eingesetzt, die aus Experimenten unter Verwendung von präzisen Tracking-Informationen stammen. Für die Evaluation der vorgeschlagenen Echtzeit-Strategie zur Tiefendatenverbesserung erfolgt ein Vergleich der Ergebnisse mit anderen aktuellen Methoden auf Referenzdatensätzen. lm Ergebnis zeigt sich, dass Rauschen und die Anzahl der flackernden Löcher signifikant minimiert und Geisterartefakte erfolgreich entfernt werden.

Weiterhin stellt die Arbeit zwei Strategien zur Echtzeit-Datenreduktion vor, mit der die von Rauschen und Artefakten bereinigten bildern unterbrechungsfrei übertragen werden können. Dies gilt insbesondere für die in vielen Anwendungsbereichen eingesetzten Mehrkamera-Konfigurationen, die entsprechend Bildströme mehrerer Kameras gleichzeitig liefern. Das entwickelte Datenreduktionsverfahren fungiert als weiterer Vorverarbeitungsschritte für die Übertragung von Szenen, die aus mehreren Blickwinkeln aufgenommen werden und daher große Mengen von Daten über den Aufnahmeort enthalten. Ausführliche Tests zeigen, dass die Reduktionsstrategien erfolgreich die Übertragung in einem Netzwerk mit niedriger Bandbreite ermöglicht.

Acknowledgements

I would like to thank first the Almighty for giving me enough courage, knowledge, and skill to accomplish this thesis work. Without his blessing, it would have been difficult to complete the required tasks within the specific time span.

This dissertation could not have been possible without the support of many people. First and foremost, I would like to show my gratitude to my supervisor, *Professor Dr. Oliver Staadt*. I thank him from the bottom of my heart for believing in my abilities and supporting me with his vast knowledge and expertise. He was not only my academic supervisor, but taught me different things about life in general, the culture, history, and management as well. I was never worried whenever I was stuck while working, because I knew that he would be there with great intellect, expertise and a friendly smile. Without his moral and academic support, it would have been difficult to accomplish this task.

I would also like to thank *Professor Dr. Renato Pajarola* for his valuable and important suggestions and guidance in my research work. I enjoyed the time we worked together and learned a lot about image processing methods as well. I like to thank also my colleagues at the Visual Computing and Computer Graphics Groups for making the work environment rewarding. I am especially thankful to *Christian Scheel* and *Anton Jirka* for being not only colleagues but also such wonderful friends. I am also grateful to *Dr. Christian Rosenke* and *Dr. Martin Luboschik* for their incredible support and valuable suggestions in my work. Furthermore, I like to express my gratitude to my Marie Curie ITN DIVA project colleagues for the incredible time I spent during the tenure of the project.

I am grateful to my parents and my sister for their love and intelligence. I thank them for always inspiring me to learn something new, to broaden my mind and to explore different aspects of life. It certainly helped me to build up my mind to dig deeper and learn new things. I am indebted to them forever. I would like to dedicate this masterpiece to my wife *Sharmin Chowdhury* and my son *Sherjeel Ahyan Tariq* for their encouragement and love that gave me the strength to do this thesis work. I am grateful to them for being so much supportive and understanding in my difficult days.

This work was funded by the *European Union FP7 Marie Curie ITN "DIVA"* under *REA Grant Agreement no. 290227* and by the *German Research Foundation (DFG)* with the research grant no. *STA 708/5-1 "SMOOTH"*.

Contents

Abstract **v**

Zusammenfassung **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Depth acquisition methods 3

1.3 Depth enhancement methods and existing challenges 4

1.3.1 Spatial-based depth image enhancement 5

1.3.2 Temporal-based depth image enhancement 5

1.3.3 Spatial and temporal based depth image enhancement 6

1.3.4 Research questions 6

1.4 Objectives and constraints 7

1.4.1 Objectives 7

1.4.2 Constraints 8

1.5 Outline and contributions 9

1.5.1 Background and related works 9

1.5.2 Novel depth image enhancement strategy 9

1.5.3 Depth noise extraction and visualization 11

1.5.4 Camera data reduction strategy 11

1.5.5 Use cases of this work 12

2 Background and related work **15**

2.1 Background 15

2.1.1 Depth images 15

2.1.2 Depth acquisition methods 15

2.1.2.1 Passive depth acquisition 16

2.1.2.2 Active depth acquisition 17

2.1.3 Least squares optimization 18

2.1.4 Least Median of Squares 18

2.2 Related work 20

2.2.1 Depth noise removal by scene depth completion and inpainting like methods 20
2.2.2 Depth noise removal by domain information 21
2.2.2.1 Noise removal using spatial domain information 21
2.2.2.2 Noise removal using temporal domain information 28
2.2.2.3 Noise removal using both spatial and temporal domain information 29

3 Real-time enhancement of depth images 33
3.1 Problem statement 33
3.1.1 Noise and invalid values 33
3.1.2 Ghosting 35
3.2 Proposed spatio-temporal method 36
3.2.1 Step 1: Spatial ghosting reduction 36
3.2.1.1 Origin of Ghosting 37
3.2.1.2 Spatial Filtering 39
3.2.2 Step 2: Temporal LMS 43
3.2.3 Real-time processing 48
3.3 Results and discussion 49
3.3.1 Evaluation using real-world depth data 50
3.3.2 Evaluation using depth data with synthetic degradation 54
3.3.3 Limitations 57
3.3.4 Future work 58

4 Depth noise extraction and visualization 59
4.1 Problem description and motivation 59
4.2 Noise extraction and analysis method 60
4.2.1 Sources of depth noise 60
4.2.2 Experimental setup 62
4.2.2.1 Description of the apparatus used in the experiment 62
4.2.3 Camera and test target calibration, and ground truth data creation . 70
4.2.3.1 Calibration of the depth camera and the test target 70
4.2.3.2 Ground truth data creation 75
4.2.4 Extraction of the test target from the ground truth image and recorded depth image 75
4.2.5 Fitting plane through the ground truth and the recorded depth image 79
4.2.6 Noise extraction and visualization of the extracted noise 80
4.3 Results and discussions 81
4.3.1 Visualization of depth images and ground truth images at varying distances 82

4.3.2 Visualization of depth images and ground truth images at varying viewing angles . . . 82
4.3.3 Visualization of depth images and ground truth images at varying light sources . . . 89
4.3.4 Observations from the experiments with Kinect V1 and V2 sensors . 89
4.4 Future works . . . 93

5 Reduction of transmission data 95
5.1 Problem description and motivation . . . 95
5.2 Related works in camera data reduction . . . 97
5.3 Data reduction by efficient degradation and enhancement of color frames . 98
5.3.1 Proposed method . . . 99
5.3.1.1 Degradation process . . . 99
5.3.2 Enhancement process . . . 102
5.3.2.1 Non Key Frame (NKF) enhancement: method 1 . . . 102
5.3.2.2 NKF enhancement: method 2 . . . 103
5.3.3 Experimental results . . . 103
5.3.4 Discussion, limitations and future work . . . 109
5.4 Data reduction using display model and light field geometry . . . 110
5.4.1 Data reduction strategy . . . 110
5.4.2 Experimental setup and procedure . . . 112
5.4.3 Results, discussions and future work . . . 114

6 Use cases of our proposed strategies 117
6.1 Utilization in telepresence applications . . . 117
6.2 Utilization in efficient 3D reconstruction of a scene . . . 119
6.3 Utilization in e-learning applications . . . 123

7 Conclusion and future work 127

List of Acronyms 129

Appendices 145

A Matlab codes for generating ground plane and visualising in 3D 147
A.1 Codes to create binary images, extract test targets and plot them in 3D . . . 147
A.2 Matlab codes for fitting planes to the captured depth image . . . 148
A.3 Matlab codes for extracting noise and mapping it onto the ground truth image149

Publications 151

Bibliography 153

Chapter 1

Introduction

1.1 Motivation

With the ongoing innovations and evolution in computer vision technologies, the realization of intelligent and automated applications are becoming more feasible in different areas, such as in autonomous vehicle industry, surveillance, virtual reality, remote collaboration, e-learning, interactive 3D scene modeling, gaming, industrial automation, forensic analysis, and robotics. Most of the applications in these areas are built upon the idea that they would be able to function and interact with the real world autonomously by being able to meticulously analyze a scene in real-time and correctly identify the objects within it. To achieve such a goal, a crucial step involves scene understanding where computer vision technologies play a vital role. Images, either in two- or in multi-dimensions, are the fundamental materials in computer vision for understanding a scene. With the introduction of multi-camera or multi-view systems for depth estimation of scene objects and with the progress in three-dimensional depth-sensing technologies, depth information of the objects in a scene is becoming more convenient to extract. Depth information is highly quantifiable in scene understanding since it contains detailed information (e.g., position, distance) of the objects in a scene. Depth information, generally, is obtained from stereopsis where the scene features are projected onto two cameras that are placed at a distance from each other (mimicking the human eye positions), and then, the depth information is extracted by using triangulation methods. However, depth estimation by triangulation methods relies heavily on finding the accurate corresponding points for matching the right and left images which is a well known but still a challenging problem.

With the innovations in camera technology, new three-dimensional depth-sensing devices, such as the LIDAR scanners, light field cameras, structured light devices, and Time-of-Flight (ToF) devices are becoming available. While high-end depth-sensing devices such as the LIDAR scanners and light field cameras are capable of generating depth images with high accuracy and comparatively less artifacts (e.g., missing depth, holes, flickering, depth inhomogeneity and alike) compared to consumer devices such as, structured light sensors and ToF devices, they are quite expensive and often difficult to operate due to their size and weight. Consequently, both the communities - academics and industry, have inclined towards easily available and low priced devices such as the structured light devices and ToF devices. Although these devices capture a scene with reasonable resolution and speed, the depth data exhibits a substantial amount of artifacts. The artifacts generated by these consumer depth cameras come in different forms. One common artifact appears as randomly distributed holes over the surface of the objects of a scene; especially, where the depth discontinuity occurs, i.e. near the edges of objects. Often, such artifact is spread over the temporal domain and causes the flickering artifacts, meaning the random holes appear and disappear at random locations on the object's surfaces over the successive frames. Moreover, in case of dramatic or drastic movement of the objects in a scene, we often perceive ghosting artifacts on the depth frames when post-processing is applied to the depth frames. Hence, the depth images from consumer depth cameras usually need further enhancement otherwise, they cannot be used in various crucial real-world applications. For example, in object tracking applications, the noise on an object's surface greatly deteriorates the tracking performance and in forensic analysis of a crime scene, where every detail is vital, a poorly reconstructed scene might hamper the recovery of correct information. Likewise, in telepresence or e-learning systems, low-quality 3D scene limits the sensation of a natural presence of the remote users to the local site.

The primary focus of this thesis is to enhance the quality of the depth data captured by the low-cost depth cameras; namely to reduce the artifacts from the resulting depth images and subsequently, the secondary focus is to reduce the amount of data required from multiple cameras (in case of a multi-camera setup) for smooth transmission of the captured data. In respect to that, we propose a new framework to compute good-quality depth images at interactive speed along with a data reduction strategy to aid uninterrupted transmission of the data. Our main contribution is the development of a new real-time depth image enhancement filter that fuses the spatial and temporal information of depth images simultaneously for stabilizing and enhancing the distorted depth data. Therefore, we suggest a composition of a novel depth outlier detection method and a real-time spatio-temporal filter. Besides this, for a better understanding of the noise characteristics of the depth sensors and eventually to optimize our depth enhancement method, we propose to develop a noise visualization and analysis procedure where we create ground truth data using position tracking information and then compare the recorded test data with the ground truth data to extract the noise. Moreover, since our objective is to improve the overall processing and enhancement strategy of the depth images, we maintain the industry requirement for ensuring the requirements of real-time interactive content that recommends less memory usage and reduction of data transmission time [1]. This recommendation is placed for ensuring smooth data transportability from one location to another. To this end, we devise a data reduction method that works as a preprocessing step before processing the captured data. Often, multi-camera setup is used for capturing a scene for visualizing a 3D reconstruction of the scene from arbitrary viewing angles; hence, a data reduction method would facilitate smooth transition of the large amount of captured data from the capture location to the processing location.

In the following, firstly we briefly discuss different methods of depth data acquisition and highlight their respective advantages and flaws. We discuss these methods and their respective attributes, because there is a direct impact of these acquisition methods on the quality of the captured depth images. Then, we discuss briefly the concept of spatial and temporal filtering approaches and the idea of fusing spatial and temporal components of depth images to attenuate the artifacts found on the depth images. Finally, we present the objectives and challenges of this thesis as well as the outline and contributions.

1.2 Depth acquisition methods

Existing depth acquisition methods can be divided into two main categories - *contact-based* and *contact-less* approaches. Contact-based approaches, as the name suggests, require some form of physical contact with the object being scanned/captured and usually delivers high quality 3D model. However, since they require direct physical contact they might not be suitable for certain computer vision applications. On the other hand, contact-less approaches do not need direct physical contact with the target objects and hence are being used in a wide variety of computer vision applications. Contact-less approaches can be further divided into categories - passive and active depth acquisition approaches where the former one uses two cameras to acquire depth using triangulation methods [1] and the later uses one camera and a projector to acquire depth [1].

Passive depth acquisition approach is based on passive triangulation method that basically reproduces the human stereovision by placing two cameras placed at a certain distance from each other. In this approach, binocular disparity [1] is used to estimate the actual depth between the objects and the cameras. However, it requires accurate detection of the projection points, which is a well-known yet challenging correspondence problem [1]. Moreover, it also requires very precise calibration of the cameras and careful setup of the instruments; otherwise, it generates invalid or missing depth information [2]. More details about this approach are stated in Chapter 2.

Active depth acquisition approach, on the other hand, is based on laser or structured light techniques [1]. In this case, a camera and an emitter, that projects a pattern or a light of specific wavelength to the scene, are used to obtain the depth information. Structured light devices and ToF cameras use this active approach to estimate the depth of a scene. While these devices are active range sensors and they are being utilized in many computer vision application due to their low-cost and easy availability, they often produce invalid or missing depth values due to reasons such as specular surface, occlusion and alike [1]. This leads to artifacts such as random holes over the surface of the scene objects and flickering. More details about this approach are stated in Chapter 2.

Because of these artifacts, many crucial 3D computer vision applications demand further enhancement of the depth images so that those applications can deliver accurate and precise output by using depth images with minimum or no artifacts. Thus, the primary aim of this thesis is to combine the spatial and temporal aspects of depth images for removing or minimizing the artifacts and hence, elevate the overall quality of a captured scene. Besides this, while capturing the detailed depth information and color information of the objects of a scene, these cameras yield massive amount of data that needs to be transported or transmitted to the processing location. Hence, the secondary focus of this thesis is to eliminate input data that does not contribute to the final output.

1.3 Depth enhancement methods and existing challenges

There has been quite a lot of research pursued by the scientific communities where researchers formulated the problem of the depth image enhancement with different approaches, including but not limited to diffusion-based enhancement [3–5], energy minimization [6–10], exemplar-based enhancement, spatial-neighborhood-based enhancement, temporal information based enhancement and so on. Reformulating the depth enhancement problem eventually resulted in a wide variety of enhancement approaches. Here, we will focus on the spatial- and temporal-neighborhood-based solutions since these methods have proven to be yielding moderate-quality output with reasonable processing speed and with low computational complexity. Of course, the other methods have their own benefits and flaws, such as energy minimization based solutions generate comparatively accurate and plausible output, but their optimization process are often difficult to implement and they have numerical instability along with large computation time. Another example could be the exemplar-based methods that show great potential in enhancing depth images where the structural continuity of a scene is preserved and the missing depth values are recovered with plausible values. However, the success of these methods in enhancing depth images greatly depend on the presence of color texture in all regions of the accompanying color images. Lack of color texture on a smooth surface eventually causes deterioration of performance for these methods.

In the existing literature of depth image enhancement using domain (either spatial or temporal, even a combination of these two) information, we came across three basic categories of methods. Among them, one category of methods uses the spatial domain information available locally within the depth map and potentially the accompanying color image, whereas another category uses the history of temporal information within a continuous sequence of images to estimate the depth values for the current depth image; the remaining other category uses both the spatial and temporal information to estimate the plausible depth values for the scene regions with missing depth information. A brief discussion on these three categories of enhancement methods is presented below.

1.3.1 Spatial-based depth image enhancement

In the spatial-based depth image enhancement methods, neighboring pixel values and other information around the affected depth pixel with artifacts are used to estimate plausible and valid depth values for the affected depth pixel. In this approach, typically a single depth image is considered while obtaining the neighborhood information of an affected depth pixel. Some solutions using the spatial-based enhancement use accompanying color image as a guidance image to estimate valid depth values for filling the holes caused due to invalid or missing depth values. The solutions based on this approach typically are suitable for static scenes and even when they are applied to dynamic scenes, the whole processing is done offline. Besides, when solutions rely on guidance color image, lack of color information in certain regions of the color image causes performance degradation of the depth enhancement.

Most common solutions using this approach of depth enhancement use spatial or range kernel filters such as median filter, bilateral filter and sometimes a combination of different filters. Apart from these filters, there are interpolation and extrapolation methods that are also used for depth image enhancement. However, although the filtering methods typically generate good quality output for static scenes, most of them show a tendency to blur the image, introduce artifacts around boundaries, and produce noisy edges. There are also inpainting-based spatial methods, which works fine to generate good quality output, but most of them work only for static scenes. Reconstruction-based methods also generate good-quality output; however, they suffer from long computational time and high complexity that cause difficult implementation.

1.3.2 Temporal-based depth image enhancement

This approach of depth image enhancement uses motion and temporal information within successive frames to enhance a depth image. Some solutions, which use this approach, also use accompanying color image to refine the respective depth image. In this approach, the history of the depth values for the affected pixels is used to estimate plausible and valid depth values to fill the holes present at the affected pixels. Temporal based approaches generally deliver reasonably good quality output for dynamic scenes which spatial-based methods are not able to perform. Besides, they also maintain depth consistency and homogeneity on the enhanced depth images. This approach is often able to deliver the output with reasonable processing speed and its computation complexity is also usually low. However, the solutions based on this approach often suffer from latency issues because of processing a number of previous frames to generate the desired enhancement of the current frames. Often, in case of dynamic scenes, we perceive flickering artifacts on the enhanced images. Moreover, for drastic or very fast movement of the objects inside a sequence of frames, enhanced images from this approach suffer from ghosting artifacts when such scenarios are not taken into considerations. Besides this, we often observe persistent holes in one part of an output scene when the depth values for that part in the previous frames are invalid or missing.

1.3.3 Spatial and temporal based depth image enhancement

This category of depth enhancement approaches fuse the attributes of spatial- and temporal-based methods and recover the depth values using both the spatial and temporal domain information. The solutions, using this approach, take advantage of the best attributes of both the spatial-based methods and the temporal-based methods and hence deliver reasonably good quality output. However, they also inherit the flaws of both the methods which often are seen as ghosting artifacts, flickering artifacts, and delay in real-time output generation.

Hence, we propose a new method of spatial and temporal based enhancement which keeps the advantages of spatial and temporal based methods and additionally minimizes or removes the ghosting artifacts and flickering artifacts while filling the holes.

1.3.4 Research questions

While the existing challenges in depth image enhancement, mentioned above, indicate which are the most important issues that need to be addressed for achieving good quality depth images with reasonable processing speed and computational complexity, below we formulate the following set of research questions which depicts these challenges more precisely. These questions are directed towards the capability of an enhancement method in addressing the existing challenges.

- Is the depth enhancement method capable of enhancing both static and dynamic scenes in real time?
- Does the method only remove the holes or can it also remove other artifacts such as flickering, motion artifacts (ghosting)?
- Does the depth enhancement method perform online or is it applied offline on the input data sets?
- Does the method's performance depend heavily on the texture of the accompanying color image to enhance the corresponding depth image?
- Can the method fill large holes without introducing additional artifacts?
- Is the computational complexity high for the enhancement process?

- Does the enhancement processing pipeline consider how the massive amount of produced data be transmitted from one location to another?

1.4 Objectives and constraints

1.4.1 Objectives

There exist quite a lot of works to address the issues with low-cost depth cameras that use different strategies to elevate the quality of depth images by recovering the depth information in the affected region of a captured scene. Among the existing methods, some of them greatly enhance the depth images, but at the cost of low processing speed while others are more efficient in processing but their enhancement performance is poor. Moreover, there are methods that work only for stationary scenes and some others work very well for stationary scenes and perform poorly for dynamic scenes. Apart from that, many of the existing depth enhancement methods' working pipelines do not take into account the massive amount of data that the camera (or cameras, in case of multi-camera setup) generates.

The objective of this thesis is to overcome these limitations, such as ghosting, flickering, processing delay, which are seen when both the spatial and temporal domain information are used to enhance the depth images of a scene. Our purpose is to develop such a method that not only would remove the mentioned artifacts but also would not introduce additional artifacts like those that some of the existing methods do. Besides, we would also like to make sure that our method has very low computational complexity and its implementation is relatively simple so that it can be applied to various applications with minimum effort. Moreover, we would like to support real-time processing speed for our approach so that it can be used in various applications that demand such speed. While developing our approach, we would also like to focus on using minimum usage of data so that we can put a minimum load on processing; to do that we opt to use only the depth image and not any guidance image. An important goal of our work is to support both static and dynamic image enhancement so that different applications can benefit from our approach. Moreover, our goal is also to process the images online so that it can work in real-time, unlike a few depth enhancement methods that opt to process the data in off-line. For a better understanding of the characteristics of the noise generated by the depth sensors, we would also like to visualize and analyze the noise generated by the depth sensors. To do that, we would like to create ground truth using our tracking setup and then compare the test data with the ground truth to extract the noise. Moreover, we would also like to ensure better transportability of the generated data so that they can be transmitted from one location/device to another via a regular speed network.

To that end, we propose a new real-time enhancement filter that fuses the spatial and temporal information of depth images simultaneously for stabilizing the distorted depth data. Here, we suggest a composition of a novel depth outlier detection method and real-time spatio-temporal filter to achieve good quality depth images. Moreover, we design a tracking setup to get the location of the camera and the test object in world coordinates, and then we create ground truth images; which we later use to extract noise from the captured test data set. We develop an analysis and visualization procedure of the sensor noise for a better understanding of the noise characteristics. The outcome of the noise analysis could essentially be used to optimize our depth enhancement method. In addition, we also aim to address the industrial requirements for real-world applications, which imply an easy and transparent adaptability of the method and an implementation capable to perform in real-time. Moreover, our work also aims to maintain the industry requirement for lower consumption of memory and data transmission time by reducing the input camera data in case of a multi-camera acquisition setup.

1.4.2 Constraints

In this work, we choose to use the affordable and easily available active depth acquisition devices simply because currently a great number of academia and industry use such devices for their respective computer vision applications. Besides, we also did not consider contact-based depth-sensing devices because these devices require physical contact (i.e. markers on the objects) with the object being scanned; which is not feasible for many applications and impractical to survey a defined area. We opted for a real-time solution and maintained a certain level of accuracy because there are quite a lot of applications that demand real-time processing speed rather than very high level of accuracy. Although accuracy is important, but many applications do not require a very high level of accuracy, instead, they are fine with normal accuracy but they would rather demand high processing speed. We also kept in mind that some applications might need more than one capturing camera in which case quite a big amount of data needs to be processed and later transmitted to different locations or devices. In this case, oftentimes a reduction of input data from each camera helps smooth transfer of processed data over low bandwidth network. We consider low-bandwidth networks in such case because there might be locations (e.g. remote geographical area, areas affected with natural disaster and alike) and situations (e.g., a depth camera, mounted on a robot at a disaster location, is transmitting the captured data over public mobile network to the base station for reconstructing the affected area in 3D) where a high-bandwidth network might not be available.

1.5 Outline and contributions

This section gives the outline of the thesis, highlights the contributions with respect to the different stages of depth image enhancement process, and provides references to the articles where the results were published. Figure 1.1 depicts a pipeline of the stages of a typical depth image enhancement process. The purpose of each of the stages is described briefly with labels in the diagram of Figure 1.1. It also shows the parts of the pipeline where we have contributed. Below we state the outline of this thesis by presenting and briefly discussing about the main parts of the thesis. The main body of the thesis is separated into five parts as below:

- Background and related works
- Novel depth image enhancement strategy
- Depth noise extraction and visualization
- Camera data reduction strategy
- Use cases of our proposed strategies
    - Telepresence
    - Efficient 3D representation of a scene
    - E-learning

1.5.1 Background and related works

This part of the thesis presents a brief background study about the topic discussed throughout the thesis and analyze the existing works pursued in the scope of depth image enhancement. In Chapter 2, firstly we introduce the fundamental topics related to depth image processing and least square optimization which are crucial for this thesis work and then, we categorize different existing methods according to their filter type usage, usage of guidance color image, processing speed, real-time support and so on. We also discuss about the respective advantages and flaws of these methods.

1.5.2 Novel depth image enhancement strategy

We present our main contribution in this part of the thesis. Chapter 3 discusses and illustrates the underlying reasons behind the existence of holes, flickering artifacts, and ghosting artifacts. Then it introduces our novel depth outlier detection method and real-time spatio-temporal filtering. It also includes the related algorithm and illustration of how invalid depth values and unstable valid depth are identified as outliers and later removed in order to attenuate the holes and flickering artifacts. The different illustrations

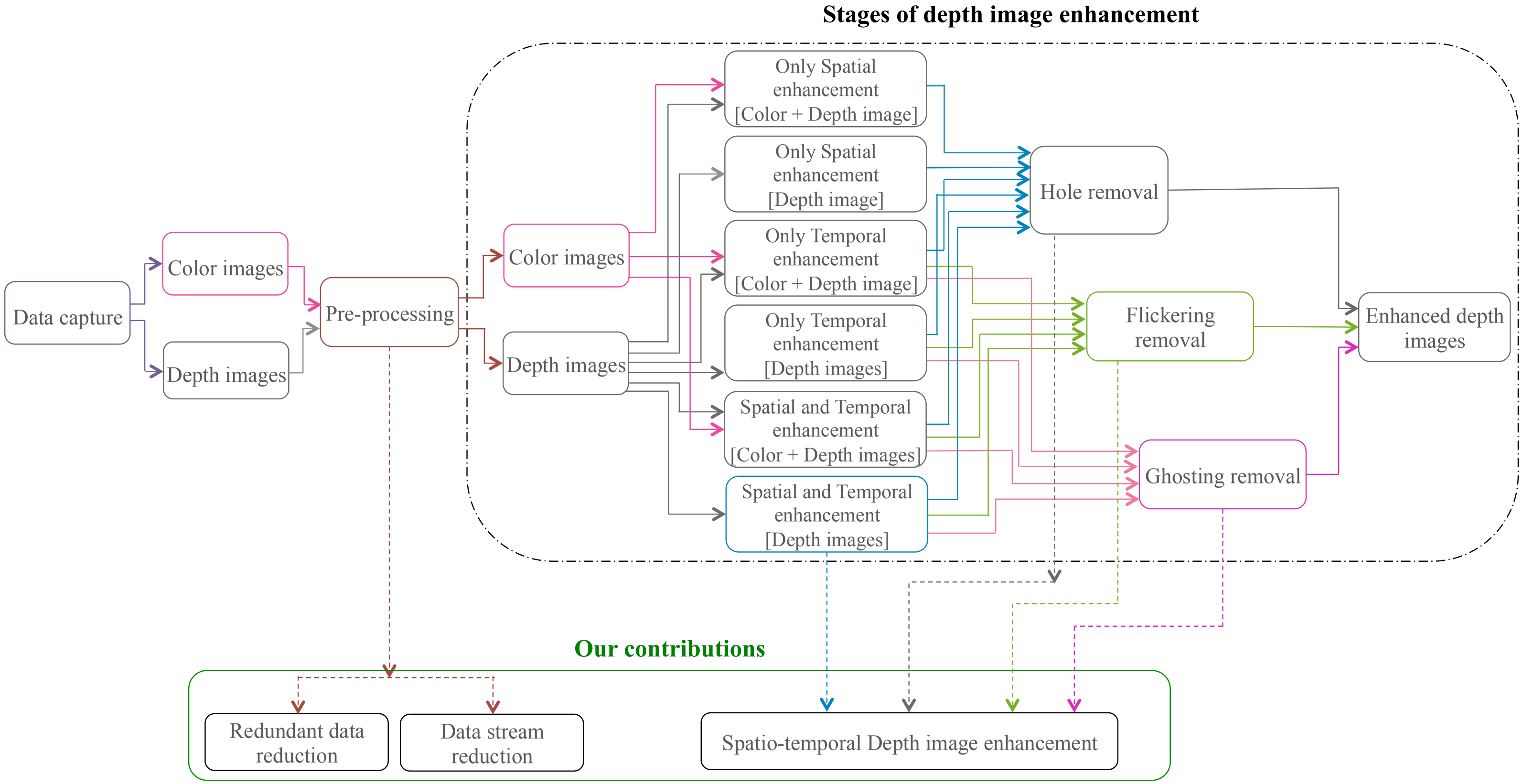

Stages of depth image enhancement
Data capture
Color images
Depth images
Pre-processing
Color images
Depth images
Only Spatial enhancement [Color + Depth image]
Only Spatial enhancement [Depth image]
Only Temporal enhancement [Color + Depth image]
Only Temporal enhancement [Depth images]
Spatial and Temporal enhancement [Color + Depth images]
Spatial and Temporal enhancement [Depth images]
Hole removal
Flickering removal
Ghosting removal
Enhanced depth images
Our contributions
Redundant data reduction
Data stream reduction
Spatio-temporal Depth image enhancement

Figure 1.1 – Pipeline of typical depth image enhancement process and our contributions.

in this chapter show how we minimize or remove the holes, stabilize the valid depths and yield ghosting-artifact-free depth images that can later be used in crucial computer vision applications. This chapter highlights our contribution in achieving a real-time and robust depth image enhancement method that performs quite well both for static and dynamic scenes. The results of this chapter have been published in the following articles:

- Islam, ABM T.; Luboschik, M.; Jirka, A. & Staadt, O., gSMOOTH - A Gradient based Spatial and Temporal Method of Depth Image Enhancement, Computer Graphics International (CGI)'18, Bintan, Indonesia, Pages 175-184, 2018.
- Islam, ABM T.; Scheel, C.; Pajarola, R. & Staadt, O., Robust Enhancement of Depth Images from Depth Sensors, Computers & Graphics Journal, Volume 68, Pages 53-65, 2017.
- Islam, ABM T.; Scheel, C.; Pajarola, R. & Staadt, O., Depth Image Enhancement using 1D Least Median of Squares, Computer Graphics International (CGI)'15, Strasbourg, France, 2015.
- Islam, ABM T.; Scheel, C.; Pajarola, R. & Staadt, O., Robust Enhancement of Depth Images from Kinect Sensor, IEEE Virtual Reality Conference, Arles, France 2015

1.5.3 Depth noise extraction and visualization

This chapter presents our work that we pursue to visualize and analyze the noise that a depth sensor yields on the surface of depth images. Chapter 4 shows how the sensor noise of a depth camera is related to the distance of the objects from the camera, the viewing angle of the camera and the lighting condition of the scene. The experiments conducted in this chapter therefore have three parameters – the distance of the objects from the camera, the viewing angle and the lighting condition. With the data obtained from this experiment, ground truth data is generated which is later used to extract noise from the captured depth data. Detail description of the experimental setup, hardware tools, camera calibration method, object tracking tools for generating the ground truth data and extracting the noise is also described in this chapter. Moreover, a brief description of the software tools developed to visualize the noise or distortion is also presented in this chapter. Results from this chapter could potentially be utilized to optimize the depth image enhancement processing pipeline stated in Chapter 3.

1.5.4 Camera data reduction strategy

This chapter presents the secondary focus of this thesis that is the reduction of camera data. There are certain computer vision applications where multiple cameras are used for capturing a scene from various viewing angles. In such multi-camera setups, first, not all the camera data from each camera is used for the final output. Hence, some parts of the camera data can be discarded. And, second, although our proposed depth data enhancement framework does not depend on the accompanying color images, many computer vision applications use the color images in addition to the enhanced depth images to reconstruct a colored 3D model. Hence, we propose two data reduction strategies – one for reducing the input data in case of a multi-camera setup and another for the color images that are captured in parallel with the depth images by the depth cameras. Chapter 5 firstly discusses briefly some existing data reduction strategies and then proposes two data reduction strategies for multi-camera setup and color image data reduction respectively. The results of this chapter have been published in the following articles:

- Islam, ABM T. & Staadt, O., Bandwidth-Efficient Image Degradation and Enhancement Model for Multi-Camera Telepresence Environments, Proceedings of the 10th European Conference on Visual Media Production (CVMP), 2013.
- Adhikarla, V. K.; Islam, ABM T., A.; Kovacs, P. T. & Staadt, O., Fast and Efficient Data Reduction Approach for Multi-Camera Light Field Display Telepresence Systems, Proceedings of the EEE 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2013.
- Islam, ABM T.; Ohl, S. & Staadt, O., Multi-Camera Acquisition and Placement Strategy for Displaying High-Resolution Images for Telepresence Systems, Eurographics Posters, 2013.

1.5.5 Use cases of this work

This chapter is presented as an extension of the works pursued in the previous chapters and typical uses cases of the proposed approaches. Chapter 6 introduces three typical use cases where we can use our proposed works. First, an illustration of a typical telepresence system is presented that shows how our proposed real-time enhancement can be used in telepresence systems. Second, how the enhanced depth images from our proposed strategy can improve the output of 3D scene reconstruction is explained and experimental results are presented. Third, an illustration of an e-learning environment is presented and then how enhanced depth image can improve the sensation of 3D presence is demonstrated. We published the following articles that showcase that these applications can potentially benefit from the enhanced depth images.

- Islam, ABM T.; Scheel, C.; Imran, A. S. & Staadt, O., Fast and Accurate 3D Reproduction of a Remote Collaboration Environment, Virtual, Augmented and Mixed Reality. Designing and Developing Virtual and Augmented Environments, Springer International Publishing, 2014.

- Islam, ABM T.; Flint, J.; Jaecks, P. & Cap, C. H., A proficient and versatile online student-teacher collaboration platform for large classroom lectures, International Journal of Educational Technology in Higher Education, 2017.
- Scheel, C.; Islam, ABM T. & Staadt, O., An Efficient Interpolation Approach for Low Cost Unrestrained Gaze Tracking in 3D Space, ICAT-EGVE - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments, 2016.

Chapter 7 concludes the thesis and elaborates on possible lines of future work.

Chapter 2

Background and related work

2.1 Background

2.1.1 Depth images

A depth image or depth map $d(x, y, z)$ is an image that contains information relating to the distance of the surfaces of scene objects from a viewpoint; here x and y are the 2D positions of the image sensor pixels which refers to the row and column of an image and z refers to the distance to the target object from each image sensor pixel. Since a depth image contains a lot more detailed information about the objects of a scene, it can be used in a wide variety of applications such as 3D reconstruction of a scene, robotics, autonomous vehicle industry, security, object tracking and alike.

Depth images are captured with various available depth sensors, such as structured light devices (Microsoft Kinect, Intel RealSense, ASUS Xtion and alike), ToF sensors, LIDAR scanners and so on. These devices use different methods to capture scene objects. Below we discuss briefly different depth acquisition methods.

2.1.2 Depth acquisition methods

There exist quite a few depth-sensing approaches, along with their respective advantages and flaws, such as depth from motion, stereo imaging, structured light or ToF [1]. These approaches can be divided into two main categories - contact-based and contactless techniques. Contact-based approaches require some form of physical contact, either the markers are to be placed on the objects being scanned or the scanning devices need to be in physical contact, and they are able to reconstruct high quality and precise 3D model of a scene. However, due to the requirement of physical contact, these approaches might not be suitable for many computer vision applications and unrealistic to scan a defined region. Contact-less approaches, as the name implies, do not require such direct physical contact with the objects being scanned and hence are being used in a wide variety of computer vision applications. Contact-less approaches can be further divided into two categories - passive and active depth acquisition approaches where the former one uses two cameras to acquire depth using triangulation methods [1] and the later uses one camera and a projector to acquire depth [1]. Figure 2.1 shows a diagram depicting the basic differences between these two approaches. In the following, we briefly describe the passive and active depth acquisition approaches.

2.1.2.1 Passive depth acquisition

Depth acquisition techniques that are based on passive triangulation method basically follow the stereopsis or stereo vision which reproduces the human stereo vision by placing two cameras placed at a certain distance (i.e. baseline) from each other. The left image of Figure 2.1 shows an illustration of passive depth acquisition setup. In passive stereo depth acquisition, binocular disparity (the difference in retinal position between the corresponding points in the two images) is used to estimate the actual depth between the objects and the cameras. However, this approach requires accurate detection of the projection points, a well-known yet challenging correspondence problem [1], which are obtained by feature matching and hence are affected by shadows or texture patterns. Moreover, this approach requires very precise calibration of the cameras and careful setup process; otherwise, even for very small issues in calibration and synchronization, it generates invalid or missing depth information. Such artifacts also occurs in case of the absence of camera overlap, featureless surfaces, sparse information for a scene object such as shrubbery, unclear object boundaries [2] and so on.

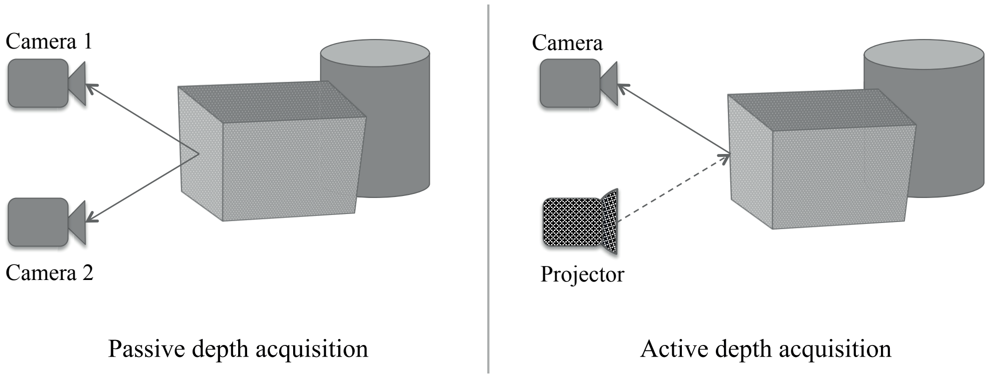


Figure 2.1 – Diagram showing the basic differences between Passive and Active depth acquisition techniques.

2.1.2.2 Active depth acquisition

In contrast to the passive depth acquisition approach, active approach, based on laser or structured light techniques [1], reduces the dependency on texture to deal with feature correspondence pairs. In this case, one of the cameras in the setup of Figure 2.1 is replaced by an emitter that projects a pattern to the scene. By doing so, the camera is able to distinguish the projected pattern from the rest of the elements, regardless of their texture. Thus, the projected pattern generates a group of features that may be detected in the recorded intensity image. ToF cameras also use an active approach of depth measurement where depth is calculated by measuring the phase difference between emitted and reflected infrared signal [11]. While both the Structured light devices and ToF cameras are active range sensors and they suffer from mis-calibration issues, they are more widely utilized for a variety of purposes due to their low-cost availability in the commercial market with factory calibration settings [2]. Moreover, these depth cameras provide more reliable and robust 3D geometry information of real world objects than the stereo-based systems [11].

However, and despite the efforts in redesigning the illumination patterns and factory calibration settings, different artifacts occur, in case of structured light devices, when the projected pattern becomes too weak compared to the background light due to a wide range of issues such as ambient light [1], external active illumination source interference [2], active light path error caused by reflective surfaces, occlusion, erroneous light pattern detection in dynamic scenes, depth offset for non-reflective objects and others [2]. ToF cameras also produce artifacts that originate from the electronic noise, dark noise and photon shot noise of the camera sensor [12]. These artifacts eventually produce invalid depth measurements. Regardless whether the active depth acquisition is performed with structured light sensors or by ToF sensors, the obtained depth information of a scene often contains invalid or missing depth values. This leads to artifacts such as randomly distributed holes over the surface of the scene objects and sometimes, those holes are perceived as flickering. Moreover, ghosting artifacts are also perceived when fast movements occur inside a scene. Due to the presence of these artifacts, many computer vision applications demand further enhancement of the depth acquired by these sensors so that those applications can deliver accurate and precise output by using depth images with minimum or no artifacts. Since the aim of this thesis is to remove or significantly reduce these artifacts and hence improve the depth image quality, we proceed toward a depth enhancement approach where we identify the invalid depths as *outliers* among the valid values that are the *inliers*. We use least squared optimization approach to maximize the probability of detection and removal of the outliers (i.e. the invalid depth values) and replace them with valid depth values.

2.1.3 Least squares optimization

Least squares is a linear regression procedure to determine the best-fit line to a set of data points [13]. The basic problem is to find the best-fit line, see Equation 2.1, given that, for $n \in 1, ..., N$, the pairs (x_n, y_n) are observed. The least squares method can be generalized to find the best fit in the form stated in Equation 2.2.

$$y = ax + b \tag{2.1}$$

$$y = a_1 f_1(x) + ... + a_K f_K(x) \tag{2.2}$$

In case of least squares method, instead of finding the best fit line, we could find the best fit given by any finite linear combinations of specified functions. Hence the general problem is: given functions $f_1, ..., f_K$, find values of coefficients $a_1, ..., a_K$ such that the linear combination (see Equation 2.2) is the best approximation to the data. The Equation 2.1 involves two free parameters that specify the intercept (a) and the slope (b) of the regression line. The least square method defines the estimate of these parameters as the values which minimize the sum of the squares (hence the name least squares) between the measurements and the model (i.e., the predicted values) [14]. This amounts to minimizing the expression in Equation 2.3 which is the error ε associated with the function, of Equation 2.1, for a given data $\{(x_1, y_1), ..., (x_N, y_N)\}$. The goal of the least square optimization process is to find values of a and b that minimize the error ε.

$$\varepsilon(a, b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2 \tag{2.3}$$

Least squares methods possess the ability to explain a problem with simplicity and have a widespread applicability in various applications. Moreover, they are the maximum-likelihood solution and, if the Gauss-Markov conditions apply, the best linear unbiased estimator [14]. However, despite these qualities, least squares methods have a crucial drawback that is high sensitivity to outliers (i.e. extreme observations). This is a consequence of using squares because squaring exaggerates the magnitude of differences (e.g., the difference between 25 and 15 is equal to 10 but the difference between 25^2 and 15^2 is equal to 400) and therefore gives a much stronger importance to extreme observations i.e. the outliers. This problem is addressed by using *robust* techniques (such as the least median of squares) that are less sensitive to the effect of outliers [14].

2.1.4 Least Median of Squares

The Least Median of Squares (LMS) [15] is a statistical technique for robust regression of a p-dimensional sample set (x_i, y_i), which minimizes the median of the squared residuals

${r_i}^2$ over a set of estimates θ in $\mathbb{R}^p$. Here, r_i and θ are expressed as in Equation 2.4 and Equation 2.5:

$$r_i(n) = y_i - (x_{i1}\hat{\theta}_1 + \cdots + x_{i(p-1)}\hat{\theta}_{p-1} + x_{ip}\hat{\theta}_p) \tag{2.4}$$

$$\theta = (\theta_1, \theta_2, \cdots, \theta_p)^t \tag{2.5}$$

Basically, a vector n represents a hyper plane that, in a certain way, describes the linear dependency of y_i from the $p-1$ variables in x_i. LMS is known to be robust to false matches and outliers while fitting equations to the observed dataset. Figure 2.2 shows an illustration of LMS that is applied on a dummy data set (x_i,y_i). It estimates the data points by solving the nonlinear minimization problem that is $r_i(n)$. The outliers from the dataset can be filtered out by using a robust standard deviation $\hat{\sigma}$ which uses the minimum median of $r_i(n)$.

It is worth to mention that LMS has never been used for depth image enhancement. We use it for the very first time for enhancing depth images. However, the original LMS estimator becomes computationally expensive when the dimension p of sample set (x_i, y_i) gets bigger. Since, in the scope of this thesis, we can define our observed depth values as a one dimensional set of data, we can use a one dimensional LMS which would be a good fit for the nature of the problem domain of this thesis and it would also be less time

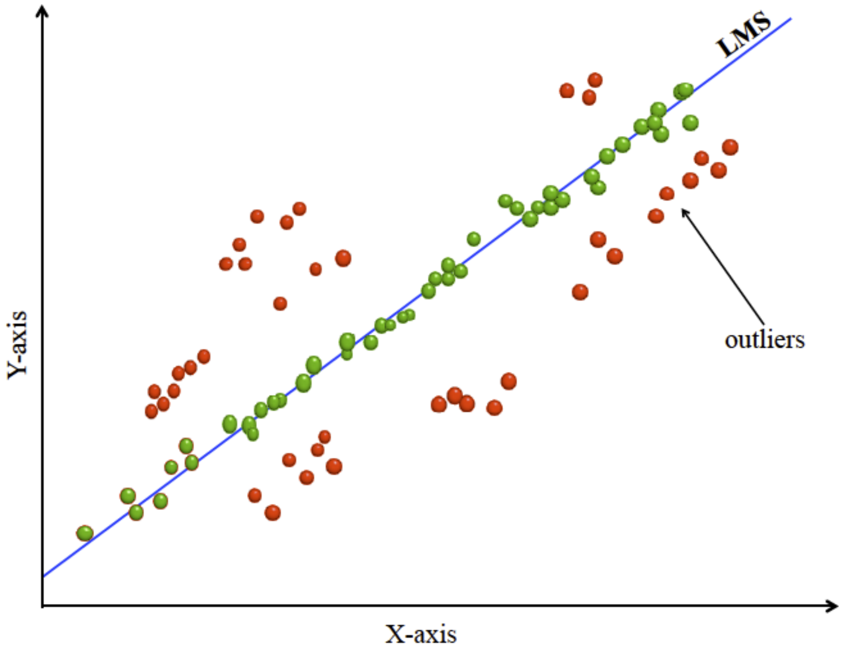


Figure 2.2 – Illustration of Least Median of Squares: the green dots represent the inliers or valid values in the dataset, the red dots represent the outliers and the thin blue line represents the Least Median of Squares regression line which goes through the inliers.

consuming than higher dimensional LMS. We choose LMS estimator over other robust estimators, such as median and least trimmed squares regression, because by using the LMS regression, we can have least expensive computation for enhancing depth images and at the same time, remove most amount of outliers with LMS than the other estimators.

2.2 Related work

There are a number of existing approaches that deal with the noise removal in depth images. These existing approaches can be divided into two main categories. First, several approaches define the depth noise removal problem as scene depth completion [2] and inpainting process where the holes can be reduced or removed by applying different strategies such as anisotropic diffusion, energy minimization, exemplar-based filling and matrix completion. Second, other approaches define the depth noise as the missing of domain information and they are based on the nature of retrieving the missing information domain that is needed for processing the depth images to fill the holes of the depth images. Here, we focus on the second category since this thesis falls into this category. We first describe briefly the outcome and limitations of the first category of these approaches and then we describe the attributes, advantages and limitations of the second category.

2.2.1 Depth noise removal by scene depth completion and inpainting like methods

Researchers, in quite a few approaches, have reformulated the problem of depth hole filling as scene depth completion and hence applied inpainting like methods [16] to fill the holes. One such method is anisotropic diffusion [3] and a few approaches, such as [4, 5], use this method to remove the noise from depth images. The approaches that use this method generally fill depth holes by extracting the edges from the accompanying color image captured from a RGB-D sensor and then by applying various diffusion methods to smooth the edge and other regions. Although the approaches based on this method yield smooth depth images in the presence of flat surface with sharp edges, their high computational cost and complexity restrict them from being used in real-time applications. The approaches based on energy minimization, on the other hand, use an energy function [6] which incorporates the characteristics of a depth image acquired via a RGB-D sensor (such as Microsoft Kinect) into the hole filling process. Approaches, such as [7–10], that use this method generally assume that a linear correlation exists between the depth and color values within a small region. These approaches generally produce smooth surfaces with sharp object boundaries, but often texture and relief information of the surface are lost during the processing [2]. Besides this, these methods also suffer from large computational overhead [8] due to the complex optimization process of energy minimization approaches; although recent computational advancements (e.g., using GPU acceleration) would facilitate them to extensively boost their computation speed.

Exemplar-based methods, in case of color image completion, generally work by copying and pasting the texture patches from the known regions of the image to complete or fill the region of interest [2]. However, since the depth images from the commodity depth sensors do not contain such level of texture and often produce smooth object surfaces, directly applying the exemplar based color image filling method is very challenging. Although the approaches, such as [17–23], which use exemplar-based filling method produce images with sharp edges, crisp surfaces and maintain structural homogeneity, often they are computationally expensive and their performance relies heavily on the availability of fronto-parallel views [2]. Few other approaches, such as [24], use matrix completion based approaches where the depth images are filled based on the idea that similar patches in a color-depth image pair lie in a low-dimensional subspace and can be approximated by a low-ranked matrix. Although the approaches using this method generate sharp edges with crisp surface, some approaches (e.g., [24]) require noisy color image as input.

2.2.2 Depth noise removal by domain information

The approaches that use domain information to remove the noise from depth images can be categorized into three types. The first type uses the spatial domain information that is locally contained within the depth map. This type of work often also includes information from accompanying color image. The second type uses temporal information extracted from a sequence of frames and use that information to remove the depth noise. The third type combines both the spatial and temporal domain information for enhancing depth images. In Table 2.1, we present the advantages and limitations of each of the three categories of depth image enhancement methods which we discussed above. It is worth to mention that the advantages and limitations differ in the degree and strength of these methods. The diagram in Figure 2.3 illustrates an overview of the existing depth noise removal methods based on the type of input data and information domain dependency.

2.2.2.1 Noise removal using spatial domain information

This category of depth noise removal methods use the depth value and other information from the spatial neighbors of a single depth image to remove the artifacts such as holes or invalid values within the current depth image. Some methods of this category also use the information from the accompanying color image to remove the artifacts from the depth image. Due to the nature of the processing attributes of this category of methods, they are mostly suitable for processing a single frame at a time or with delayed results in case of processing a sequence of frames. They can also be applied to processing a sequence where off-line processing is allowed for an application. Although these methods have the potential to generate real-time results with the help of recent advancements in hardware

Table 2.1 – A review of advantages and limitation of different depth image enhancement strategies.

Enhancement strategy	Sub-strategy	Advantages	Limitations	Related works
Spatial-based strategy	Spatial-filtering methods	• Good performance for single frame. • Easy implementation. • Potential to perform in real-time by using hardware accelerations, e.g. GPUs.	• Not always suitable for dynamic scenes. • Dynamic scenes can be enhanced only offline. • Cannot remove ghosting and flickering artifacts. • Unwanted edge smoothing.	[25–33]
	Reconstruction-based methods	• Generates clean image. • Good accuracy near object's edges.	• Dependency on guidance image. • Slow processing speed. • Complex implementation. • Not suitable for online processing.	[6–8, 34–38]
	Inpainting-based methods	• Provides output with sharp edges. • Better handling of smooth regions.	• Might fail to fill large holes. • Not suitable for dynamic scenes. • Computationally expensive.	[16, 23, 39–43]
Temporal-based strategy		• Capable of processing dynamic scenes. • Preserves depth consistency. • Potential to address flickering and blurring.	• Lower quality output. • Might not remove ghosting artifact. • Processing speed might be slow.	[44–50]
Spatio-temporal based strategy		• Capable of producing better output than temporal-based methods	• May perform at slow speed. • Might not remove ghosting artifact. • Dependency on guidance image.	[51–55]

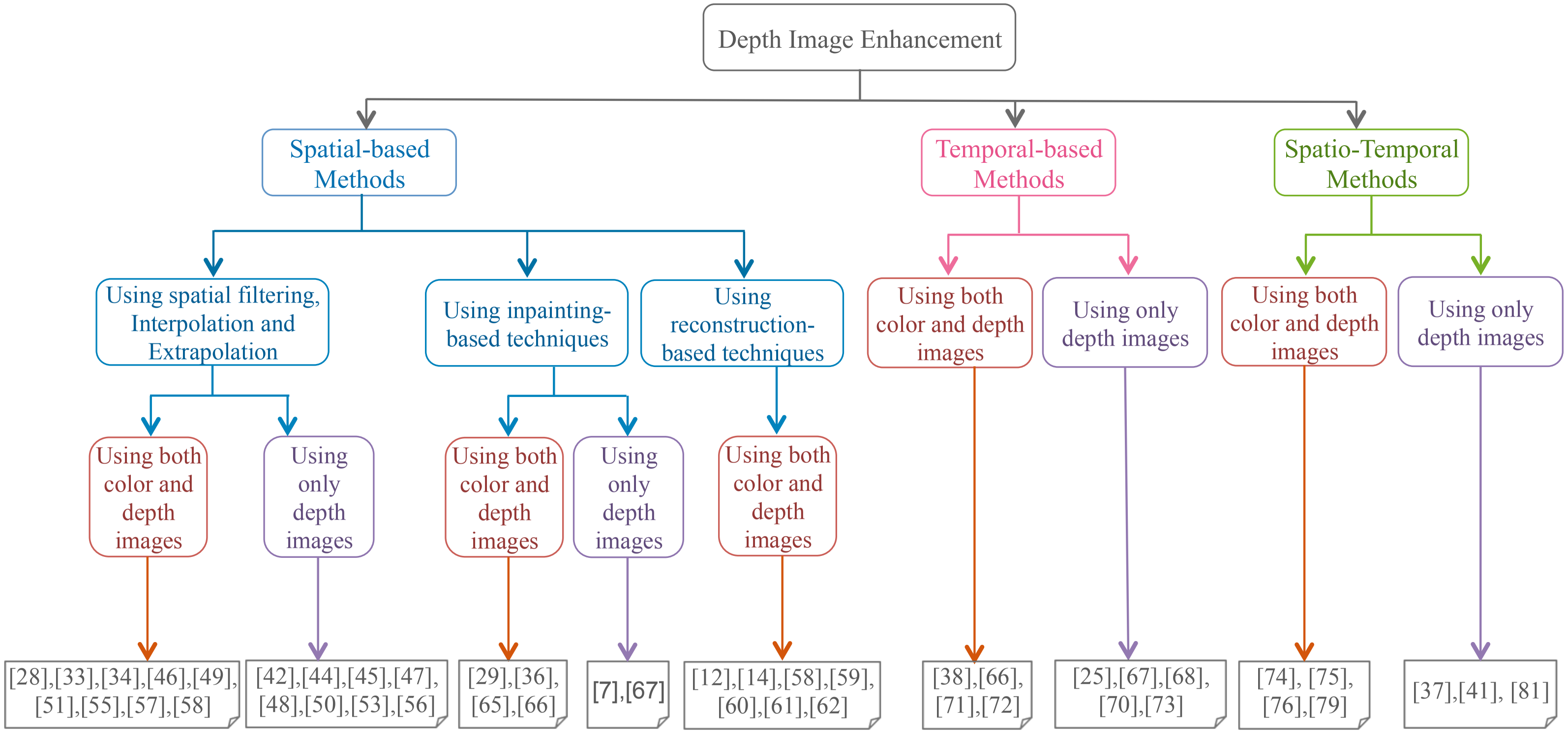


Figure 2.3 – Diagram illustrating the input data and information domain requirements for the existing depth image enhancement methods.

acceleration and optimization in their respective algorithms, such real-time solutions can be achieved only if there is no dependency on other frames [2]. Existing approaches in this category of methods can further be divided into three main groups based on the type of spatial information these methods use for removing the artifacts. Below we briefly discuss the works that falls into these three groups.

Methods based on spatial filtering, interpolation and extrapolation

There are quite a few approaches that use only spatial filters for depth image enhancement. Most of these approaches use popular filters like median filters [32,46], Kalman filters [56], guided image filters [30, 40, 57, 58], bilateral filters [25–27, 39, 55, 59, 60], and sometimes a combination of median and bilateral filters [26, 45]. Besides the type of filters used, these approaches can basically be categorized by the depth sensor type they use, real-time processing support and by the inclusion of accompanying color image data. In most of the depth image capture scenarios, either the color images are captured with the attached RGB sensor of the depth cameras or a secondary RGB camera is used to capture the color images when the depth cameras do not have one. Many of the existing depth enhancement approaches use the visual information encoded in the color image to further enhance the accuracy of the depth images. Many of these approaches, e.g. in [61, 62], in fact use the color images to elevate the sharpness and resolution of the depth images.

For instance, Chen et al., in [25], use accompanying color image to fill the holes using a region growing approach. They use a joint bilateral filter to further elevate the accuracy of the enhanced depth images. However, it fails to work well for parts where the color image contains a dark region. Yang et al. [26] also enhance depth images by using bilateral filters that generate good result for static scenes; however, due to lack of temporal information, it is not suitable for dynamic scenes. Camplani et al., in [27, 28], also use a joint bilateral filter that evaluates pre-detected foreground areas and edge-diffidence maps to integrate depth and color information. However, it is applicable only to static scenes. Shen et al. [29] propose another method using bilateral filters that assumes different depth layers by separating the scene into a static background and several dynamic foreground objects. Later, they combine different RGB-D noise models to determine the label of each depth layer and fill the holes considering the fact that only the neighboring pixels that are on the same depth layer contribute to filling the central pixel. Their output outperforms the output from [28].

He et al. [30] use a guidance image to enhance the depth image. They use a linear time guided filtering approach where the content of the guidance image is used to generate the resulting image. Their method performs fast and is able to preserve the sharpness of the edges since it transfers the structures of the guidance image into the resulting image. Some other notable methods that also use a similar guidance image approach are [40, 57, 58, 63]. However, most of them are not suitable for dynamic scenes and a few of them yield blurry object boundaries.

Yang et al., in [31], use a different approach of applying a bilateral filter for depth image enhancement. They fill the holes based on the depth distribution of the neighboring pixels. To do so, at first they label each hole and then dilate each labeled holes to get the value of the surrounding pixels. Later, they use a cross-bilateral filter to elevate the accuracy of the output. Another method proposed by Nguyen et al. [64] also uses a cross-bilateral filter to fill the holes in the warped image. They use the propagation of the directional depth information that is based on camera calibration to fill the holes caused by disocclusion from 3D warping [64]. While this approach generates good results, it works only for the holes occurred due to transformation and warping.

Min et al., in [65], propose to use a new way of using the information from color images to enhance the corresponding depth image. They use a weighted mode filter and a joint histogram of the color and depth image pair. At first, they analyze the color similarity between the target and the neighbor pixels to obtain a weight value that is then utilized to count each bin on the joint histogram of the depth image [2]. Their method also includes temporal information for achieving a temporally stable depth video. Daribo et al. [66] use another weighted filter, a weighted Gaussian filter, unlike a weighted mode filter in [65], to enhance the depth images. They basically apply this rather simple to implement filter by considering the distance to the contours. Here they apply smoothing close to object boundaries but avoid filtering the smooth areas in the depth image [2]. Another similar approach is proposed by Chen et al. [67]. However, here they use an average filter rather than a Gaussian filter as seen in [66]. Here they use an adaptive approach by taking into account the edge and directions that eventually helps to preserve the sharpness of the edges and avoid the smoothing the textured areas.

There are few other notable approaches, e.g. [68, 69], that use cross-trilateral median filter and multilateral filter respectively to fill the holes in depth images. However, these works are suitable for the depth data that are estimated by stereo correspondence that is not covered in this thesis. Although they produce reasonable output, occasionally they yield blurry depth images.

Quite a few depth image enhancement methods use interpolation and extrapolation techniques to fill the holes in the depth images. For instance, Garro et al. [70] propose a segmentation-based depth image enhancement method. Since this method requires accurate alignment of the objects inside the color and the depth image, it uses advanced segmentation methods [71] that combine depth and color information when the image is not particularly highly textured to identify the surfaces and objects in the color image. Subsequently, the low-resolution depth image is projected on the segmented color image and later interpolation is applied to the resulting image. Although this method produces good quality output, it's high dependency on precise registration between color and depth image causes occasional failure when the registration and segmentation are not done perfectly. Xu et al. [33] also use an advanced segmentation method, but unlike the one in [70] which is based on graph cuts [71], they use watershed color segmentation [72] for correctly aligning the color and depth images. Although their method yields output without any blurring, the segmentation is computationally expensive.

Another method that uses interpolation technique is proposed by Atapour-Abarghouei et al. [73]. They use a grammar-inspired non-parametric interpolation approach that uses a segmentation step to redefine and identify the holes into a dozen of completion cases. Subsequently, they propagate the depth pattern into hole regions according to the individual cases. Although for regular-sized holes it performs quite well, its performance highly depends on accurate segmentation that does not always occur and it also fails for large holes. Maimone et al. [32] also use interpolation method for removing holes. At first, they use a GPU-accelerated median filter and then they apply interpolation for filling the holes. This approach generates smooth depth frames, but it occasionally produces wrong interpolated values.

A few other approaches use extrapolation technique to enhance the depth images. For example, Po et al. [74] use a multi-directional extrapolation method to fill the holes. This method uses the neighboring pixel texture features to estimate the direction in which extrapolation is to take place, rather than using the classic horizontal or vertical directions that create obvious deficiencies in the completed image. They propose sets of nine directions to fill the holes so that there is a higher possibility for the completed holes to match the texture or structure of the background and the surrounding objects. Other notable methods using such strategy are [75,76].

Methods based on reconstruction techniques

The approaches that use reconstruction based approach for depth enhancement basically define the problem of hole filling as an energy minimization problem. Although most of these approaches use the autoregression model or Markov Random Fields for depth enhancement, they use different objective functions that originate from their respective regularization terms.

For instance, Yang et al. [7] propose an adaptive color-guided depth image recovery method where they utilize an auto-regressive model to define the problem of hole filling as a problem of minimizing the model's prediction errors. For improving the accuracy and stability of their depth enhancement strategy, they also apply a parameter adaptation strategy which they use for processing each pixel. A similar color-guided approach has been proposed by Garcia et al. [34] where the hole filling performance highly depends on the accurate registration between the color and the depth image pair.

A rather different approach, than [7,34], of energy function for a depth image recovery model is proposed by Liu et al. [9]. Here, for building the energy function the authors assume that a linear correlation exists between depth and color values in small local neighborhoods. They also propose to use a regularization term along with the energy function for attenuating the noise and sharpening the object boundaries. In a rather similar approach, Chen et al. [6, 35] use a regularization term, along with their energy function, which includes a joint-bilateral and a joint-trilateral filter. The joint-bilateral filter is utilized to integrate the structure information and the joint-trilateral filter is adapted to the noise model of the depth camera used for depth acquisition.

Wang et al. [36] use a trilateral constrained sparse representation (SRn) approach of depth hole filling which considers the intensity similarity and spatial distance between a reference patch (in the color image) and the target patch (in the depth image). This method is based on [37] which uses a locally regularized representation that ignores the effects of geometric distance and position of the target and reference pixels in the depth and color image pair. Wang et al. improve the output by including the SRn method into it.

Other notable works in this category are pursued by Sheng et al. in [8] and Yang et al. in [38]; here, in the former one, the authors use a combination of joint bilateral filtering and segment-based surface structure propagation, and in the latter one, the authors use an adaptive color-guided auto-regressive (AR) model. In general, the depth enhancement methods based on reconstruction techniques suffer from large computational overhead due to their energy minimization approaches [8]. However, recent computational advancements (e.g., using GPU acceleration) would assist them to overcome this issue.

Methods based on inpainting techniques

Quite a few works in the area of depth enhancement adapt the popular inpainting method that has been primarily used in enhancing color images. Although most of the works using this approach generate a reasonable output, most of them are computationally expensive and are not suitable for real-time applications. For example, the approach [23] by Criminisi et al. use a structure-guided inpainting technique [77] which generates reasonable results but occasionally suffers from blurring and loss of fine details due to its diffusion process [2]. Moreover, it is mainly used in the depth data acquired through stereo correspondence that is not covered in this thesis. Telea et al. present another image inpainting method [16] that uses both the depth and color data for depth enhancement. However, it exhibits noisy object boundaries due to not considering both the spatial and temporal information among the pixels and fails for large holes.

Another approach that uses both the color and depth images is proposed by Qi et al. [39]. Here they use a fusion-based inpainting method where the color image is used mainly for locating the object boundaries and hence, achieve nice results with sharp object edges. While filling the holes in the depth images, they use a non-local filtering strategy that considers the geometric distance, and the depth and structure similarity in the color image. Liu et al. [40] also use the information from color images to enhance the corresponding depth image; however, they adapt the fast marching method from [16]. To achieve better output than [16], they incorporate a post-processing method by using the color-image-guided technique from [30]. As a result, the output shows better sharpness near the objects' edges. There are a few other notable approaches, e.g. [41–43], which use the inpainting method in combination with the information from the accompanying color image. Among these methods, in [42] a GPU-based anisotropic diffusion-based method is used which works in real-time. Here, the anisotropic diffusion is applied to ensure the accurate alignment of the object boundaries in the color and depth image pair.

2.2.2.2 Noise removal using temporal domain information

This category of methods uses motion and temporal information from a sequence of frames to enhance the depth images. Some methods in this category also use accompanying color frames to remove the artifacts and refine the corresponding depth frames. For instance, in [44], Avetisyan et. al. use optical flow information of consecutive color frames and transfer this information to enhance the corresponding depth frames. Although their work provides real-time results and performs well in suitable cases, it does not produce satisfactory results for very noisy regions. Moreover, it occasionally generates invalid motion vectors and also suffers from ghosting artifacts when there is a sudden change in depth due to rapidly moving objects in a scene. Hui et al., in [45], estimate the optical flow of consecutive color frames in a mobile RGB-D camera setup to get an additional depth cue which then enhances the depth frames. However, instead of building a temporal filter on top of the obtained data, their method estimates additional depth cues from the flow that are then combined with the original depth images. Moreover, it works only with moveable camera setups and is not appropriate for stationary cameras. Izadi et al. [50] propose another method for moving camera setup that is called KinectFusion. Here they use a sequence of depth frames to complete the missing area while reconstructing a scene in 3D. Although their method is robust, it is only applicable to scenes with static objects.

Matyunin et al., in [46], also use the motion information, obtained through motion estimation, to enhance the depth frames. However, they present an offline approach where the filtering itself is still just spatial; the estimated motion is only employed to temporally smooth the depth images. Although their outputs are mostly plausible, they suffer near the edge of the objects. Moreover, their method occasionally generates invalid depth value when the color information does not correspond to the accompanying depth information. An online temporal method is presented by Islam et al. in [47, 78, 79] which considers the depth value history of the pixels in the temporal domain to enhance the depth frames. They mainly use a simplified but conceivably parallelizable LMS filter to enhance the depth frames. While their method exhibits satisfactory results for static scenes, it shows ghosting artifacts for dynamic scenes with fast-moving objects [44, 47].

Fu et al., in [48], use an adaptive temporal filter where the depth inconsistencies among neighboring depth frames are corrected using the correspondence between color and depth frame pairs. In their method, they observed that the depth values of the same object change from one frame to another even though the planar existence has not been changed; hence, the inconsistency in depth occurs which causes flickering artifacts. Although their method generates reasonable results by applying the temporal filter to all the regions of the scene, they do not discuss the outcome of their method in case of fast movement of objects. Sheng et. al., in [80], also offer a solution to correct the temporal depth inconsistency but by using a different approach than [48]. Here, they use an intrinsic static structure which contains the static structures of the scene. They initialize this structure on the very first frame and gradually refine it when more frames become available. They enhance the depth values by considering both the incoming input depth and the intrinsic static structure [2]; the weight of enhancement is based on the probability of the input depth value belonging to the structure [2]. This method applies the depth inconsistency correction process only on static parts of the scene in contrary to the method [48] where the enhancement process is applied to all regions.

The authors in [49] fill the holes in the depth images in two steps: first, they categorize the holes on the basis of the reason behind the occurrence of the holes and then they use the subsequent deepest neighboring values to fill the affected pixels according to the category they fall into [2]. The authors consider two alternative reasons behind the occurrence of the holes: one in which the holes are created by the occlusion occurred due to the moving foreground objects, and the other in which the holes are caused by the attribute of the objects' surface such as specularity of the surface, strong lighting condition, and other random factors. Although their assumptions might work in many cases, there might the situations where static objects can also be the reason to yield missing or invalid data in depth images which are acquired by low-cost depth cameras [2].

Therefore, the approaches which use temporal domain information generally yield decent quality output even in such cases where the spatial-domain based methods are unable to do so. When it is important to preserve the depth consistency and homogeneity in a depth sequence, the temporal-domain based approaches deliver the solution for such situations. Contrary to that, the dependence on the other frames often causes delays in processing which makes some of these approaches suitable only for offline processing [2].

2.2.2.3 Noise removal using both spatial and temporal domain information

This category of depth noise removal method combines the features from the spatial and temporal based approaches and removes the noise or holes using both the spatial and temporal information contained in depth images [28, 54]. Most of the works in this category either use some kind of filtering or use interpolation to remove the noise.

For instance, Kim et al. [51] use a joint bilateral filter that uses a combination of spatial and temporal depth enhancement process. They consider the motion flow between consecutive color images to infer information about object motion in the corresponding depth images. Although their method yields sharp and smooth depth images, it basically ignores the length of motion vector data present in the dynamic parts of the depth images and hence it only works for the static parts of the images. Whereas, Xu et al. [52] use an advanced interpolation method in combination with a motion detection strategy for depth image enhancement. Their motion detection method is based on the motion information from the temporal sequence that they use for filling up the affected regions caused due to occlusion. Then they extract the dynamic objects by using background differentials and the original images and finally, apply a four-neighbor interpolation method over the background areas before filling the body areas. Although their method preserves the sharpness of the objects' edges, the interpolation method is computationally expensive.

Camplani et al., in [28], use a joint bilateral filter to remove the holes from depth images. They use an iterative method of computing a reliability score of the neighbouring pixels' depth values to fill the holes with the most reliable neighbors around the holes. They apply the joint bilateral filter to the neighboring pixels where the weights of those pixels are determined based on visual information, depth information, and a temporal consistency map that is created to track the reliability of the depth values near the hole regions. Here, they try to increase the accuracy of the reliability values with iterative filtering and with filtering consecutive frames. Although this method generates good quality depth images which the authors compare to a popular inpainting algorithm proposed by Criminisi et al. [23] and describe their results as visually better, their method is suitable only for static scenes. Another spatio-temporal depth enhancement method [55] that uses joint bilateral filtering is proposed by Richardt et al. They use a multi-scale completion technique pursued in [81,82] to remove the holes. The resulting depth image is generated by using a joint bilateral filter and a spatio-temporal process that removes noise by averaging values from successive frames.

The approach from Camplani and Salgado [53] also uses a combination of joint bilateral filtering and a Kalman filtering to enhance the depth images. This method consists of three stages: at first an adaptive joint bilateral filter which combines the depth and color information is used, and then an adaptive kalman filter is applied to each pixel to remove the flickering artifact and finally, it fills the missing depth values by applying a 2D Gaussian kernel and by interpolating the stable depth values in the regions neighboring the holes obtained from previous stages. This method yields good results but it can fail for the image regions where color information is absent. Wang et al. [54] also use accompanying color images for enhancing the respective depth images. Their method has two stages: first, they yield a "deepest depth image" by fusing the spatial and temporal information from the color and depth image pairs, and utilize that image to remove the holes and then, the resulting depth image is further improved by using collective information of geometry and color. Their method generates good quality depth images, however, it depends highly on accurate registration of color and depth images and it can fail when the color information is not available in the accompanying color images.

More recently, Islam et al. [83] propose another spatio-temporal method which uses spatial neighborhood information from each depth pixel of a depth image to remove the ghosting artifacts and then use a sequence of frames to locate outliers with respect to depth consistency within the frame. Their method performs in real-time to enhance the depth images. They utilize an improved and more efficient regression technique LMS [84] to fill holes and replace outliers with valid depth values. The approach is capable of removing the ghosting artifacts, removing the holes and flickering and sharp depth refinement within a sequence of frames. Nevertheless, it can occasionally fail to remove holes for a specific part of an image when the previous successive depth frames do not contain any valid depth values for that part of the image [83].

Therefore, the works using both the spatial and temporal domain information for depth image enhancement presumably exploit the best attributes of both spatial-information-based methods and temporal-information-based methods, but they also acquire the flaws of those methods along with the advantages. Temporal and motion information can help to remove the blurring, jagging, and mismatched object contours that are occasionally created by spatial-based methods [2]. However, they can cause the delay in generating output that might restrict these methods to be used in real-time applications.

From the above discussion about various different depth enhancement methods and from the list of the advantages and limitations in Table 2.1, we can see that the temporal and spatio-temporal methods have the potential to be implemented for enhancing dynamic scenes. However, these methods also have certain limitations regarding their ability to properly remove the artifacts while maintaining some conditions such as relatively simple implementation, faster processing time, appropriate and minimal usage of available data, and alike. Table 2.2 presents an outline of the abilities of these methods regarding the mentioned conditions.

From Table 2.2, we can see that several of these methods can only address some of the research questions which we discussed in Section 1.3.4. None of the methods can address all those research questions which are important to process dynamic scenes captured from consumer depth cameras. For example, one method can process both static and dynamic scenes, but they are not capable of processing in real-time speed; while some others might produce good quality output, but at the cost of long processing delay. There are also some methods that generate good quality output but their processing complexity is relatively high and they are relatively difficult to implement. Besides, very few methods address the minimum data usage issue, which becomes important when the data needs to be transmitted over regular bandwidth network, but they might not be applicable for dynamic scenes and might also not support faster processing. Hence, there are yet some questions which need to be addressed to achieve better performance in removing artifacts from both static and dynamic scenes. The purpose of this thesis is to develop a depth image enhancement strategy which would essentially address all of the research questions mentioned in Section 1.3.4.

Table 2.2 – Capability of the temporal and spatio-temporal methods regarding addressing the research questions yet to be solved.

Capability of methods	**Example temporal and spatio-temporal methods method**											
	[44]	[45]	[46]	[47]	[48]	[49]	[50]	[51]	[52]	[53]	[54]	[55]
Handle both static and dynamic scenes?	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Easy to implement?	No	No	No	Yes	No	Yes	Yes	No	Yes	Yes	No	Yes
Removes ghosting?	No	No	No	No	No	No	Yes	No	No	No	No	No
Can be applied online?	Yes	Yes	No	Yes	Yes, with delays	Yes	Yes, with delays	Yes	Yes, with delays	Yes, but with delays	Yes, with delays	Yes, with delays
Dependency on color image?	Yes	Yes	Yes	No	Yes	No	No	Yes	No	Yes	Yes	No
Introduce additional artifacts?	No	No	Yes	No	Yes	No	No	Yes	No	Yes	No	No
Low computational complexity?	No	No	No	Yes	No	No	No	Yes	No	No	No	No
Removes large holes?	No	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Consider minimum usage of data?	Yes	No	No	No	No	No	Yes	No	No	No	No	No

Chapter 3

Real-time enhancement of depth images

This chapter discusses the primary contribution of this thesis. Here, we first discuss the reasons behind the most significant problems (holes, flickering and ghosting) that occur during capturing a scene with commodity depth cameras and after processing it with a temporal filter that does not take the depth frames' temporal aspects into account. Then, we propose our new method which combines the spatial and temporal information from a sequence of frames to enhance the depth images in real-time. Finally, we show the results using our self-recorded and state-of-the-art datasets. We also compare our results with the results from state-of-the-art methods used for depth enhancement. Finally, we discuss the limitations of our proposed approach and briefly write about the future work that can be done within this work. The approaches in this chapter are published in [47, 78, 79, 83].

3.1 Problem statement

3.1.1 Noise and invalid values

Depth frames acquired by depth cameras (e.g., RGB-D sensors or ToF cameras) contain artifacts such as holes (caused due to invalid depth values) and flickering (varying depth values of the same pixel over time). Such artifacts are shown in Figure 3.1 and 3.2. These artifacts occur for example, when the sensor of a Kinect cannot measure the depth information accurately. A Kinect captures both the color and depth data of a scene. In the case of Kinect V1, the depth of an object is measured by projecting light patterns from an infrared light source on the object space [85]. An infrared camera receives the reflected light pattern from the surface of the object and compares it against a reference pattern. An estimated disparity image of the objects inside a scene and corresponding depth calculations are obtained by the differences between the captured patterns and the reference pattern [27, 86]. The inaccurate depth measurements of a Kinect occur

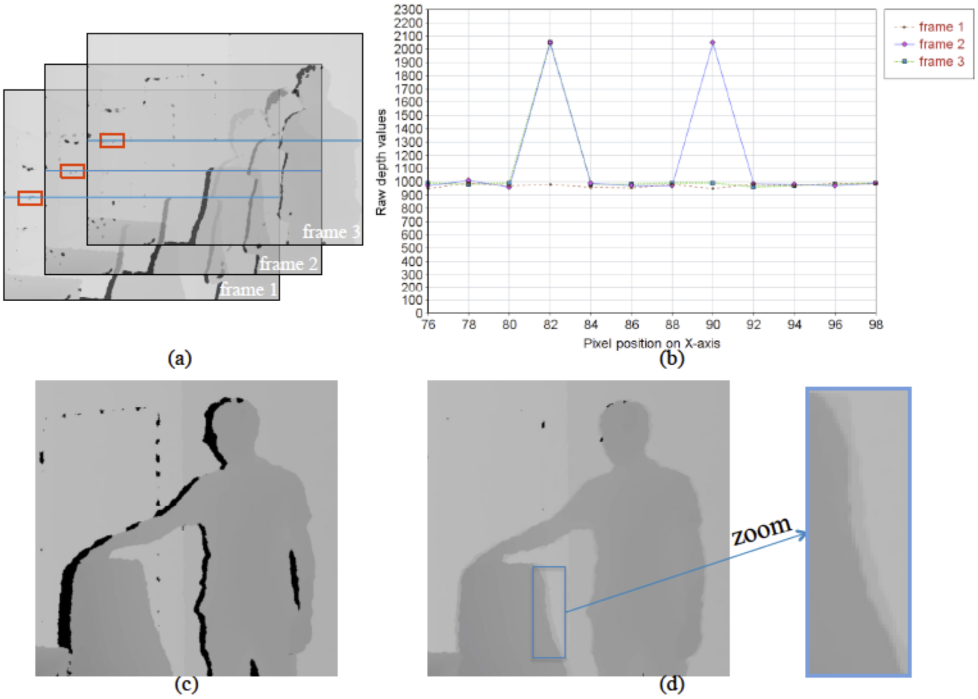


Figure 3.1 – Invalid and unstable depth values from a depth sensor on successive frames: (a) three successive depth frames, (b) the depth values along a single scan line of the rectangle area in (a), (c) a single depth frame and, the respective output (d) from method [47] showing some ghosting artifacts.

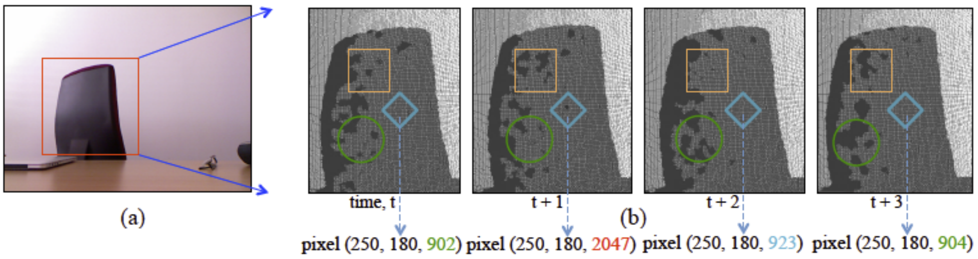


Figure 3.2 – RGB-D frames from Kinect V1 with static objects: (a) color image, (b) four consecutive depth frames of square marked area of (a). The depth values of one pixel inside the diamond marked area in (b) are depicted below the frames. We observe three types of depth value distortions: 2047 is an invalid value (reported occasionally), 923 is an outlier and the others are regular depth values fluctuating because of Gaussian noise.

in the presence of multiple reflections, strong lighting conditions, and scattering object surfaces [27, 85, 86].

Typically, the inaccuracy in the depth measurements for a depth sensor like Kinect originates from three principal sources: the depth sensor, the measurement setup and the attribute of the object's surface [85]. The sensor errors occur mainly due to the insufficient sensor calibration and flawed disparity measurement. While the inaccurate calibration parameters estimation causes the systematic error in the object coordinates of individual points, the incorrect disparity measurements influence the accuracy of individual points. Errors caused due to the measurement setup are mainly related to the lighting condition and the imaging geometry. The lighting condition influences the correlation and measurement of disparities. In strong light, the laser speckles appear in low contrast in the infrared image, which can lead to outliers in the output [85]. The imaging geometry includes the distance to the object and the orientation of the object surface relative to the sensor. The possibility of occurring erroneous depth measurement increases with increasing distance to the sensor [85]. Moreover, depending on the imaging geometry, parts of the scene may be occluded or shadowed which appear as gaps or large black areas. In Figure 3.1(c), the left side of the chair is shadowed because it is not illuminated but is captured in the infrared image. The attributes of the object's surface also affect the measurement of points. As we see in Figure 3.2(a), shiny surfaces that appear overexposed in the infrared image (the top part of the chair) impede the measurement of disparities, and hence, result in black holes in the depth image.

A major issue with the depth image from depth sensors like Kinect is the presence of pixels for which the depth values are not obtained (see Figures 3.1 and 3.2) and hence, they contain invalid depth values and appear as black holes in the depth images. Such holes occur primarily because of occlusion near object boundaries or scattering object surfaces, but they are also seen in areas that correspond to concave surfaces and, randomly, in the homogeneous image regions. Depth measurements at object boundaries are also severely affected by noise. Moreover, sharp depth transitions that occur near the object boundaries yield spurious reflection patterns that derive inaccurate depth measurements causing incorrectly aligned object boundaries [27].

The depth measurements are also prone to instability over time, which results in flickering [27]. It occurs even in the static parts of a scene (see Figure 3.2); here, we can observe the unstable nature of valid depth values for certain pixels. The plot in Figure 3.1 (b) shows that Kinects report both invalid values (see the spikes) and fluctuating valid values (see the different values marked by different shaped markers) for the same pixels in successive frames. There are several approaches to address this problem; amongst others successful are the temporal filtering techniques.

3.1.2 Ghosting

A majority of the depth enhancement methods uses a variety of filters to deal with the holes and flickering artifacts. Especially with temporal filters [47, 51], ghosting artifacts become a serious problem in dynamic scenes. Such artifacts occur near edges of fast-moving objects as the depth values at those areas change drastically from one frame to another [44, 47]. Since temporal filters combine depth values from temporally consecutive frames, the fast-changing values in such dynamic regions are often falsely recapped into incoherent depth values [44] (e.g., by interpolation). Those incorrect values along the edges are perceived as ghosting artifacts in the form of an interim depth value. Figure 3.1 (d) shows an example of a ghosting artifact near the right edge of the chair. Details about the reason behind ghosting and how we remove them are discussed in Section 3.2.1.

Here, we take this unstable nature of depth values into consideration to replace the unstable, invalid and incoherent values with plausible, valid and stable depth values.

3.2 Proposed spatio-temporal method

In this work, we propose a new gradient-based spatio-temporal Least Median of Squares (we call it gSMOOTH) approach to enhance the depth frames in real time. We consider the history of depth pixels, both in the spatial and temporal domains, over a certain number of frames and use that information to obtain stable and plausible valid depth values. We propose to use both the spatial and temporal coherence which will correct every depth pixel by incorporating values from spatial and temporal surrounding areas. Our approach is divided into two steps: in the first step, we spatially process consecutive frames using a gradient-based approach which helps to prevent the observed ghosting artifacts from the resulting final depth frames. In the second step, we apply a temporal LMS filter which generates stable and plausible valid depth values for every depth pixels. Figure 3.3 depicts the pipeline of the processing steps of our method. We depict the detailed processing steps for one pixel, marked with a yellow square, showing how that pixel is enhanced in the spatial and temporal steps of our proposed approach; see Figure 3.4.

3.2.1 Step 1: Spatial ghosting reduction

Although our temporal LMS-approach [47] (where we did not incorporate the gradient-based spatial filter) for depth enhancement yields compelling results with noise removal

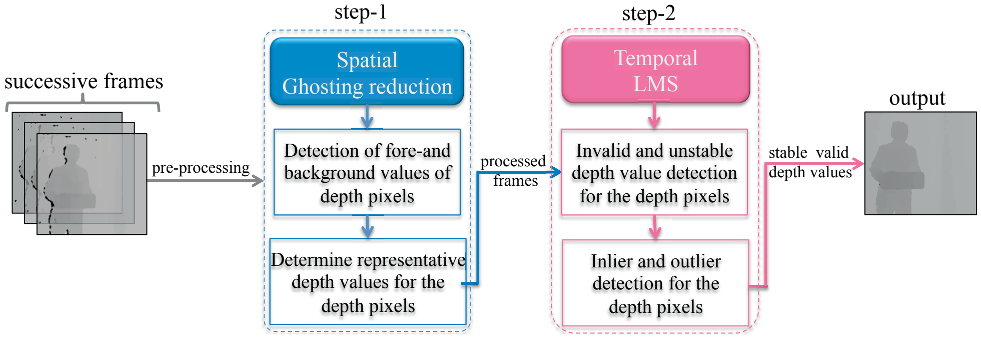

step-1
step-2
successive frames
Spatial Ghosting reduction
Temporal LMS
output
pre-processing
Detection of fore-and background values of depth pixels
processed frames
Invalid and unstable depth value detection for the depth pixels
stable valid depth values
Determine representative depth values for the depth pixels
Inlier and outlier detection for the depth pixels

Figure 3.3 – Illustration of the processing pipeline of gSMOOTH.

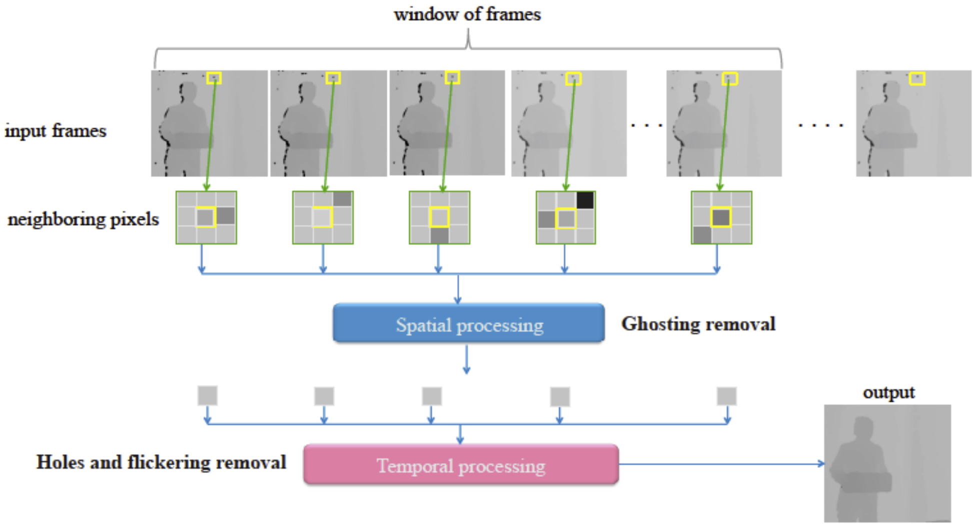


Figure 3.4 – Detail processing steps of gSMOOTH for the pixel marked with yellow square. In the spatial processing step, the attributes from the neighbouring pixels of the yellow marked pixel are used to remove the ghosting artifact which occurs on the final output for dynamic scenes. Holes and flickering artifacts are addressed for that pixel in the temporal processing step and finally the enhanced output is generated.

and invalid value treatment, it inserted ghosting artifacts (see Figure 3.1 (d)) which are a common issue in temporal filters. The aim of our work is to keep the advantages of LMS and to additionally prevent the ghosting. For this purpose, we first have to identify the origin of ghosting.

3.2.1.1 Origin of Ghosting

Like other temporal median filter, our temporal LMS-approach of depth enhancement [47] presumes a Gaussian distribution of depth values over time with the outliers (invalid and noise values) located at the Gaussian curves respective ends. Taking the *median* value of that distribution means to *select* one of the temporally most prominent existing depth values to distinguish valid values and outliers. Taking the average of the *valid* depth values yields a final depth that is close to the median. Considering rather static uniform regions, this hypothesis works fine as there is only one true valid depth that continuously evolves over time, infrequently interrupted by outliers. Figure 3.5 (a) illustrates this assumption.

Examining the edges of fast-moving objects, we find another configuration which is eventually the origin of ghosting. Instead of a single Gaussian distribution of depths, we observe a temporal abrupt change of depth resulting in two Gaussian distributions comprising the foreground and background depths respectively. Due to the persistent noise that is relative to the valid depths, noise values from the foreground or background are now located at the center of the temporal depth value distribution (see Figure 3.5 (b)).

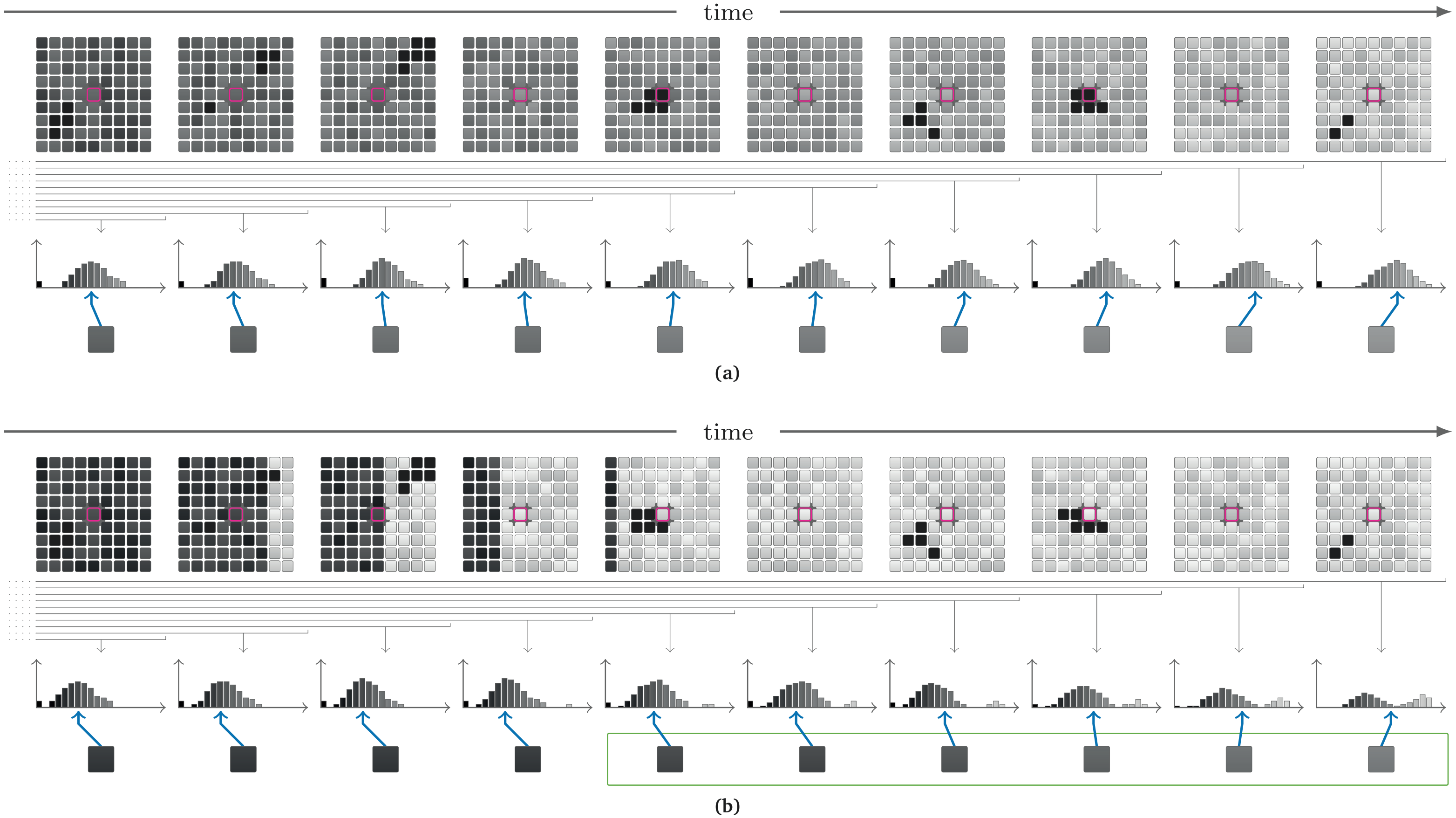


Figure 3.5 – Illustrating the origin of ghosting: The LMS depth enhancement approach [47] presumes a Gaussian distribution of raw depth values in a temporal sliding window for a single pixel (red marked center). Within a continuous depth value change (a), the respective distribution slightly shifts. Hence, the median (blue arrow) also shifts continuously and only valid depth values are finally selected for the enhancement. But in an abrupt change (b), the distribution of the pixels depth value series splits into a foreground and a background distribution. Now, as the median still shifts continuously, intermediate noise values are selected for the enhancement (see rightmost distribution in (b)). Such intermediate values are recognized as ghosting artifacts (green box in (b)) as they do not reflect the abrupt change.

Thus, the selected median is a value randomly chosen either from the foreground noise or from the background noise. Those intermediate depth values caused by noise are the origin of the ghosting artifact and the randomness of noise explains the uneven structure of the ghosting (see detail of Figure 3.1 (d)).

For the ease of explanation, we have depicted another illustration for a rather static scene, see detail of Figure 3.6, which shows sample original pixel values of a certain pixel (marked with a yellow square), continuous distribution (as more frames come into the scene) of the median values for that pixel and their final median value. As expected, when we apply a median filter, such as our temporal LMS from [47], on these frames, we can see that the median also does not vary much from one frame to another and hence, we do not see any ghosting on the final output.

For examining the effect of applying our temporal LMS [47] on a dynamic scene, we depicted another illustration (see detail of Figure 3.7) showing sample original pixel values of a certain pixel (marked with a yellow square), continuous distribution (as more frames come into the scene) of the median values for that pixel and their final median value. Here, in this dynamic scene, the object is moving from right to left. By looking at the pixel values, we can see that the foreground pixels have lower values and the background have higher values. Now, when we apply a median filter, e.g. our temporal LMS [47], on these frames, we can see that the median gradually shifts towards in between the lower and the higher pixel values. These intermediate values are perceived as ghosting. The pixels in the later frames have such values which belong neither to foreground nor to background and hence they appear as ghosting. The bottom sequence of frames shows the ghosting artifact on the pixel. A careful observation shows that the gradient of this pixel has changed (we can compare it with the gradient of the top sequence of frame's pixels) on the later frames of this sequence. It is worth to mention that our temporal LMS approach from [47] is stated later in Section 3.2.2; in that section, our gradient-based spatial filter (discussed in the next section) is already incorporated. If we apply only our temporal approach (without incorporating our spatial filter) on a dynamic scene, we would perceive ghosting on the dynamic scene (see Figure 3.1 (d)). Therefore, we proposed and developed a gradient-based spatial filter to prevent the ghosting.

3.2.1.2 Spatial Filtering

Instead of solely analyzing the history of a single depth pixel [47], the above observation motivates a prior spatial analysis including the neighborhood of the current pixel. The purpose of that step is to detect if that depth pixel is located near or directly at an edge in the depth values. With that knowledge, we can shift the later temporal LMS (discussed in Section 3.2.2) to base either on valid foreground or valid background depths without the respective noise.

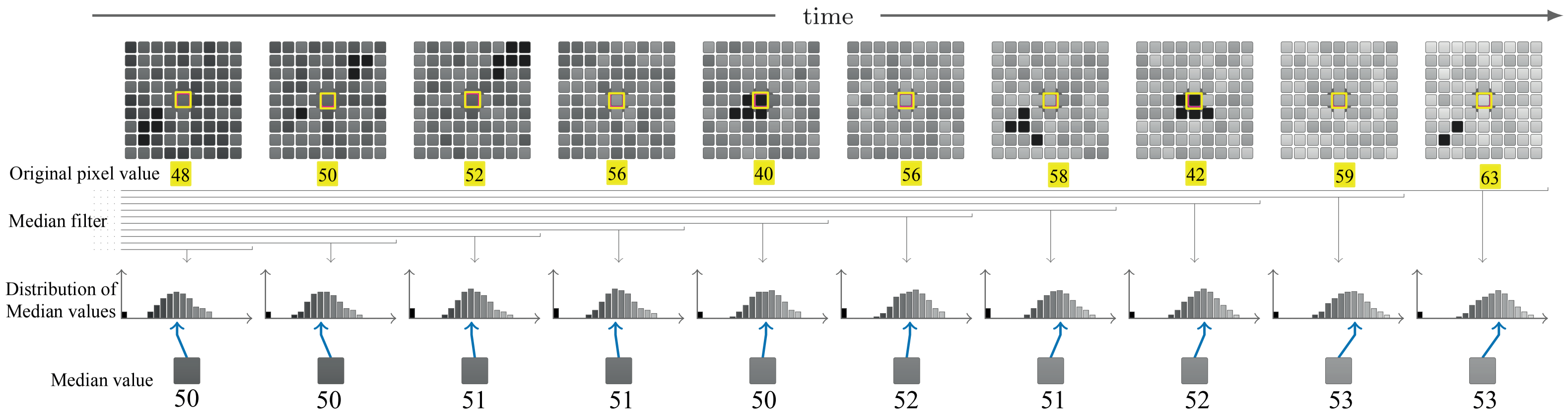


Figure 3.6 – Illustrating of the effect of applying a median filter such as the LMS approach [47] on a window of frames with rather static scene. When we apply the LMS depth enhancement approach on these frames, we can see that the final median values do not vary much from one frame to another and hence, we do not see any ghosting on the final output.

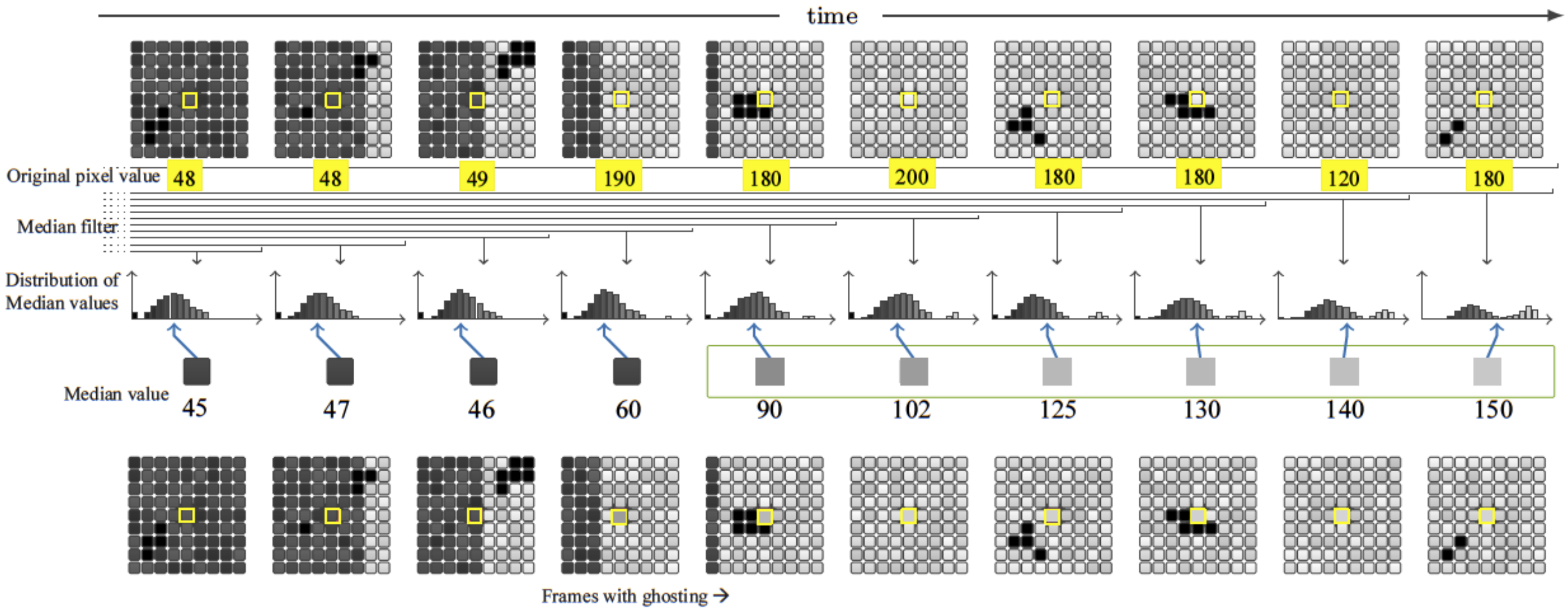


Figure 3.7 – Illustrating of the effect of applying a median filter such as the LMS approach [47] on a window of frames with dynamic scene. When we apply the LMS depth enhancement approach on these frames, we perceive a gradual shift of the median from the lower to the higher pixel value and it stays in between the lower and the higher pixel values. These intermediate values are perceived as ghosting. The bottom sequence of frames shows the ghosting by pointing at the gradient difference, i.e the beginning of the ghosting, between the top and bottom sequence of frames.

We apply our spatial analysis on a per-depth pixel basis. Here, we address the depth values by spatial references $d(x, y)$ specifying pixel coordinates (x, y). We define the spatial aspect of the surrounding area by the m-neighborhood $N_{x,y}$ of a depth pixel (x, y), which is a 2D array containing the values as stated in Equation 3.1.

$$N_{x,y}(p,q) = d(x + p, y + q); -m \leq p,q \leq m \tag{3.1}$$

Since edge detection commonly bases upon locating strong gradients, we follow the same procedure. We are interested in classifying single depth values $d(x, y)$ and propose to calculate the discrete gradients along the four main directions (0°, 45°, 90°, 135°) within the neighborhood $N_{x,y}$. Looking for edges crossing that small region, we are calculating the average gradients with respect to each pixel column, row or diagonal (see Figure 3.8).

In case we found one or more strong gradients, we know that the current pixel (x, y) is located close to an edge. That means the depth value $d(x, y)$ is affected either by the foreground, by the background or by intermediate noise. As we aim for a deterministic distinction of foreground and background values, we rely on the *majority of similar* depth values found in $N_{x,y}$. Assuming at most one edge per $N_{x,y}$, such pixels can easily be selected as they correlate with the position of the strongest gradient (see Figure 3.9). With choosing only a pixel subset $S_{x,y} \subseteq N_{x,y}$, we elect one of the two depth value distributions (foreground/background) to have a stronger influence for the pixel and its neighborhood. As we are enhancing a single pixel, we finally have to pick a single depth value $d_s(x, y)$ based on the subset $S_{x,y}$. To eliminate noise located along the edge, to simultaneously keep the strong gradient, and moreover to prevent the calculation of intermediate values, we take the median value of the subset $d_s(x, y) = \text{med}\{S_{x,y}\}$. In this way, the following temporal LMS (discussed in Section 3.2.2) bases either on foreground or on background depth values instead of intermediate noise (see Figure 3.10) and ghosting artifacts as in Figure 3.5(b) should be obsolete.

In summary, we i) use a local edge detection approach to determine whether the foreground or the background has a stronger influence on a certain pixel and its neighborhood and ii) use spatial smoothing to reduce noise. The result of this preprocessing step is

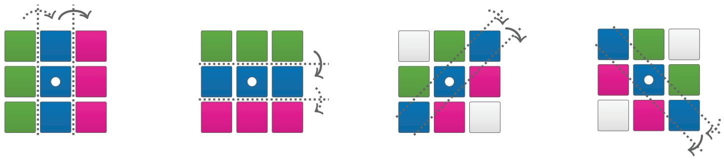


Figure 3.8 – Finding edges along the four main directions by calculating the gradients between the green, blue and red pixels average in an exemplary 3-neighborhood.

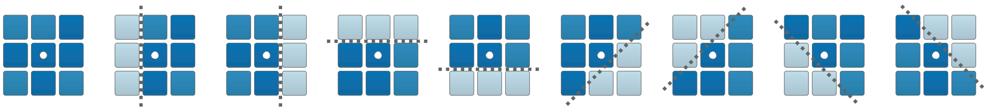


Figure 3.9 – Depending on the existence and the location of the strongest gradient (dotted line) in a 3-neighborhood $N_{x,y}$, there are 9 different subsets $S_{x,y}$ (darker blues).

a deterministically-chosen noise-reduced depth value $d_s(x, y)$ that is passed on to the temporal LMS enhancement step.

3.2.2 Step 2: Temporal LMS

We assume that a RGB-D sensor generates a sequence of depth frames. In our framework, each enhanced frame is formed based on the history of $(t - n + 1)$ frames' depth values; here t is the frame number along the time direction and n is the number of frames whose depth values are used to enhance the $(t + 1)$th frame. To enhance the (t+2)th frame, we just take the newest frame into our temporal window and remove the oldest frame from the window. The latency of our approach depends on the number of frames in the temporal window. Figure 3.11 depicts an illustration of our proposed approach.

We apply our spatio-temporal LMS on a per-depth pixel basis over the successive frames in the temporal window. Thus, the aforementioned spatial depth values $d(x, y)$ get a further reference and become $d(x, y, t)$ with t determining the frame number. Consequently, the neighborhood $N_{x,y}$, the subset $S_{x,y}$ and the chosen value $d_s(x, y)$ become $N_{x,y,t}$, $S_{x,y,t}$ and $d_s(x, y, t)$ respectively.

We obtain a candidate depth value d_c for each depth pixel (x, y) of every frame t inside the temporal window by simply taking the spatial filtering result $d_s(x, y, t)$. Considering each single pixel with x and y fixed, we have a spatially-fixed 1D array of depth values whose elements vary only along the temporal domain. Now, we apply the temporal part of our gSMOOTH approach on this 1D array of d_c^t to find a stable and valid value.

Our goal is to locate the invalid and unstable depth values (which we consider as outliers) from the set of depth values of each pixel and replace them with a stable valid depth value. An illustration of our gSMOOTH temporal LMS on k consecutive frames is depicted in Figure 3.12. Our depth enhancement strategy is based on three principle steps. For each pixel inside the sliding window (i.e. for the n consecutive frames):

- detect if there are invalid depth values.
- detect if the valid depth values are fluctuating from one frame to another.
- if the valid depth values are fluctuating, find the outliers among the valid depth values, remove them and process the inliers to get a final stable depth value. Also replace the invalid depth values with the stable valid depth value.

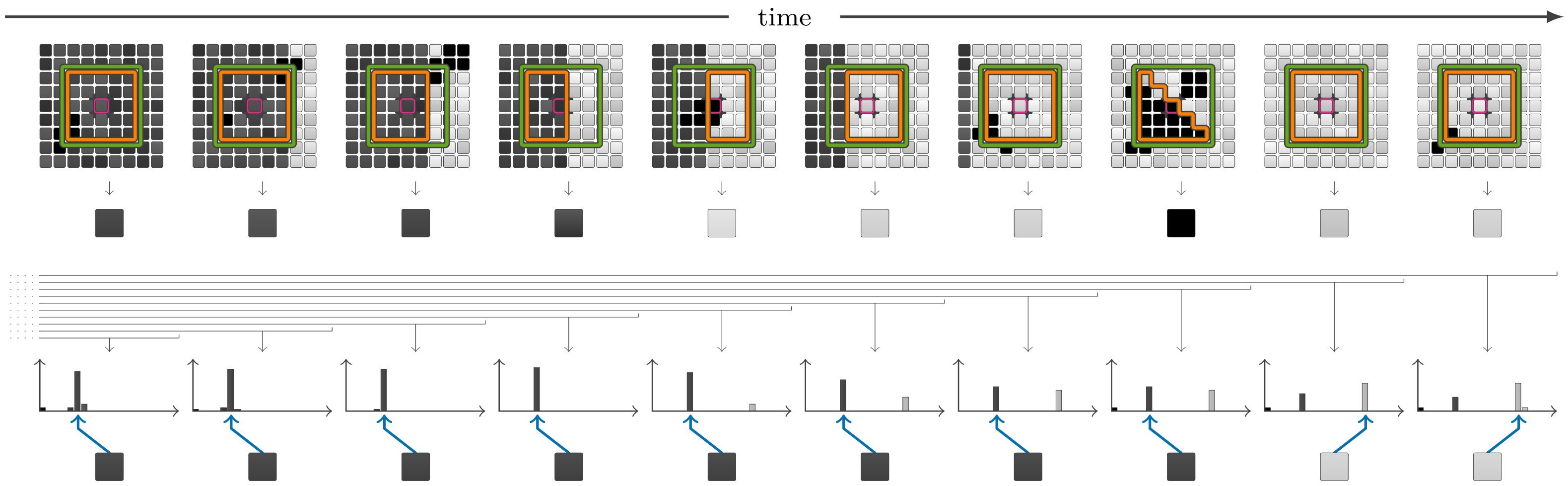


Figure 3.10 – Illustrating the principle of our approach. For a current pixel to be enhanced (red marked center) we use our simple gradient-based edge detection to determine that subset (inner orange shape) of the current neighborhood (green rectangle) that is supposed to be most similar to the pixel. The spatial median of such subsets yields a temporal series of candidate values that contains less or no noise. Thus, the distribution of depth values inside the temporal sliding windows is characterized rather by single peaks. Hence, an abrupt change in the raw depth data is reflected by an evolving change in such peaks. Compared to Figure 3.5(b) the median of the temporal distribution (blue arrow) – which is used for the enhancement – now switches abruptly. Finally, the enhanced depth values comprise no intermediate values perceived as ghosting.

We follow the definition of invalid depths as stated in [27, 86] to find the invalid depth values from the 1D array of d_c^t. To detect the valid but temporally fluctuating depth values among the d_c^t, we use the standard deviation of the valid d_c^t and then identify the elements of d_c^t as *outliers* that do not satisfy certain conditions. The invalid depth values are identified as outliers in this process. To identify the outliers among the depth values d_c^t, we propose to use the residual information among these values. We basically look for the minimum difference i.e. minimum residual among the d_c^t. Identifying the outliers from a sample data set using residual information can be performed by methods such as the least median of squares (LMS) [15]. We choose to adopt the LMS method to fit into our proposed method as it has proven to be optimal with respect to the robustness of tolerating up to 50% outliers [15].

To remove the outliers and fill the invalid depth values for a certain pixel, we calculate an LMS estimator M which identifies the stable valid values among the d_c^t such that the median of all absolute residuals r_i is minimized. Here, we use the absolute difference for getting the residuals r_i among the d_c^t; in [15], r_i^2 is used to avoid negative values. The whole calculation process is illustrated in Figure 3.13. Here, in step 1, for a set of depth values d_c^t, we calculate absolute difference for each depth value to the rest of the depth values for that pixel. Therefore, we obtain a set of residuals for each depth value of d_c^t. In Figure 3.13, the first set of absolute distances i.e. $r_0 = \{2, 21, 1145, 4, 1\}$ are calculated for the first depth value 902, r_1 for the second depth value 904 and so on. In step 2, we calculate the medians from each set of residuals r_i and then, in step 3, we calculate the LMS estimator M which is the minimum of the medians. Equation 3.2 shows the formula to calculate M.

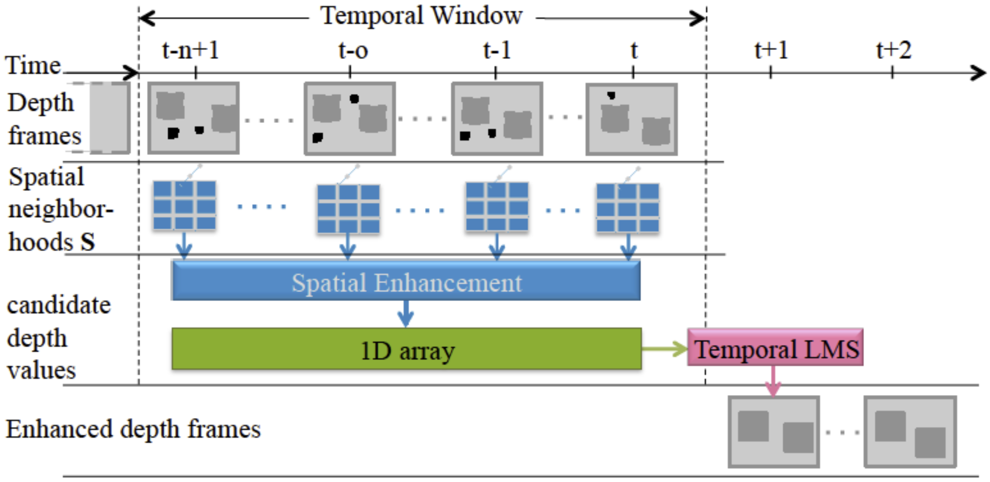


Figure 3.11 – Illustration of Spatio-temporal LMS. The temporal window contains a number of frames whose depth values are used to enhance the $(t+1)th$ frame. The spatial neighborhoods **S** are shown for one depth pixel of the frames.

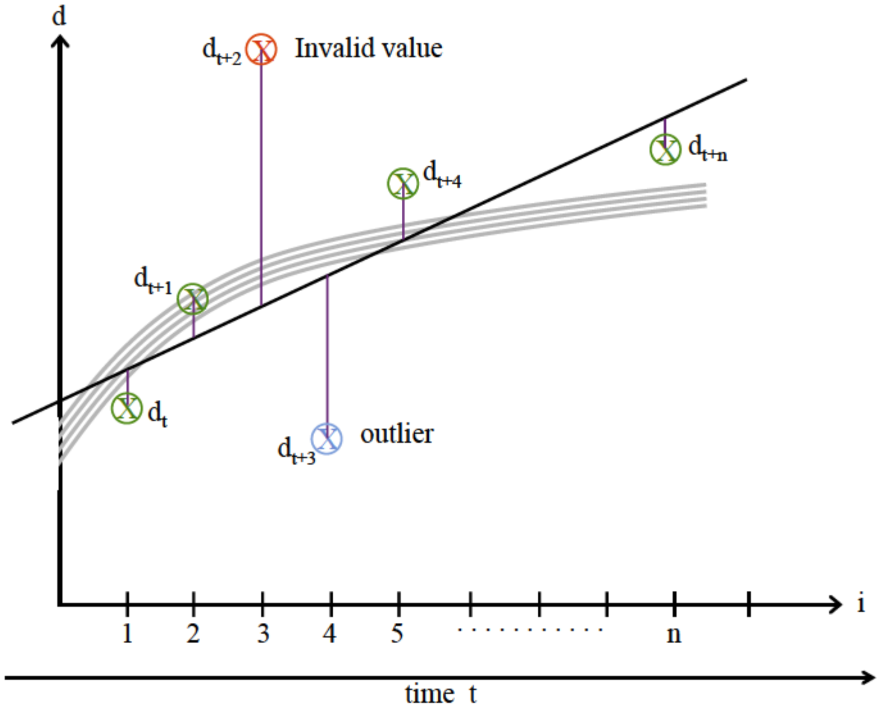


Figure 3.12 – Illustration of our gSMOOTH temporal LMS on one depth pixel in a sliding window. The thick gray line shows the real distance of a scene object over time. The checked circles represent depth values d_{t+i-1} obtained in n consecutive frames. The thin black line approximates the gray line by calculating the estimator M i.e. by minimizing the median of the absolute distances to the depth values, visualized by vertical purple lines. The red marked *invalid value* at $i = 3$ and the turquoise marked *outlier* at $i = 4$ are detectable by our temporal LMS.

After that, we retrieve the corresponding depth value whose index matches the index of M. Here, in step 3, we can see that the value of M is 2; which originates from the second set of residuals r_1. As r_1 originates from the second depth value 904, we use the value of M and the retrieved depth value (904) for the rest of the process of our temporal LMS. We calculate the standard deviation $\hat{\sigma}$ and a constraints c_i, to determine if a certain depth value is an inlier or outlier, according to the formula stated in Equation 3.3. After we obtain inliers and outliers, we finally obtain the stable valid depth value by taking the average of the inliers.

$$
\begin{aligned}
M &= \min_{i} \left(\operatorname{med}\{r_i\}\right) \\
&= \min_{i,j} \left(\operatorname{med}\{|d_c^i - d_c^j|\}\right) \qquad (3.2) \\
i, j &\in [n-t+1, t], i \neq j
\end{aligned}
$$

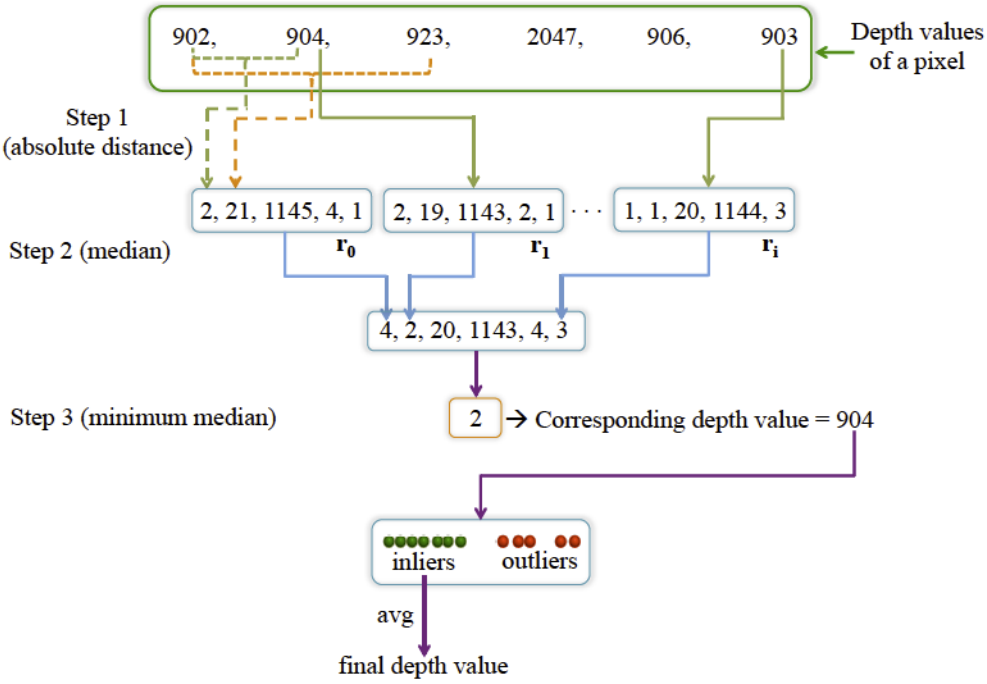


Figure 3.13 – Illustration of gSMOOTH temporal LMS on one depth pixel – the values on the top green box are the raw depth values obtained for one pixel over the frames inside the sliding window.

$$\begin{aligned} \hat{\sigma} &\cong 1.4826 \left[\frac{n+4}{n-1} \right] M \\ c_i &= \begin{cases} 1, & \text{if} \quad r \leq 2.5\hat{\sigma} \\ 0, & \text{if} \quad r > 2.5\hat{\sigma}. \end{cases} \end{aligned} \tag{3.3}$$

The constant 1.4826 in Equation 3.3 is a coefficient to achieve the optimal efficiency in the presence of Gaussian noise, $(n+4)/(n-1)$ is used to compensate the effect of using a small set of data and the constant 2.5 is used with $\hat{\sigma}$ because in a Gaussian situation there will be very few residuals larger than $2.5\hat{\sigma}$; for a detailed explanation of the coefficients, please look at pages 17, 29, 44, 131, 132 and 202 in [84]. However, this process needs quite some time to traverse for all the depth values of each pixel for these n frames. Because, in Figure 3.11, we can see that to obtain the absolute distances r_is, we need to check each depth value to the rest of the depth values to get the r_is which takes quite some time. Therefore, to reduce the calculation of r_is, we adapt a fast data traversal procedure to fit our temporal LMS.

The fast data traversal procedure is given in Algorithm 1. It shows the process of enhancing *one* depth pixel over n successive frames. Here, we show the processing of the candidate depth values d_c^t. Note that, we use an advanced median calculation strategy than [15] whose effectiveness is explained in [84].

Algorithm 1 Spatio-temporal LMS for the values of one depth pixel over a window of `n` frames

```
procedure
    variables
        i, j, count, n, r, M
        cDepthSrt[n]; sorted candidate depth values (these values are the resulting depth values denoted by $d_s(x,y)$ in the gradient-based spatial filter discussed in Section 3.2.1.2)
        maxVal; a value ≥ reported max depth value
        nDepth[j]; calculated new depth values
        FinalMedian; median of new depth values
        inliers, validStableDepthVal
    end variables
    M ← maxVal
    j = 0
    for i = 0 to $\frac{n-1}{2}$ do
        r = cDepthSrt$\left[i+\frac{n}{2}\right]$ − cDepthSrt$[i]$
        if r ≤ M then
            M = r
            nDepth[j] = $\frac{\text{cDepthSrt}\left[i+\frac{n}{2}\right]+\text{cDepthSrt}[i]}{2}$
            j = j + 1
        end if
    end for
    FinalMedian = MEDIAN of nDepth[0...j]
    $\hat{\sigma}$ = 1.5 $\left[\frac{n+4}{n-1}\right]$ M
    count = 0
    for i = 0 to $n-1$ do
        if |FinalMedian − cDepthSrt$[i]$| ≤ 2.5 × $\hat{\sigma}$ then
            count = count + 1
            inliers = inliers + cDepthSrt$[i]$
        end if
    end for
    validStableDepthVal = $\frac{\text{inliers}}{\text{count}}$
end procedure
```

3.2.3 Real-time processing

Our algorithm has the advantage of processing each depth image pixel (x,y) independent of all the others, thus it is easily parallelizable. In order to support real-time applications, we implemented gSMOOTH on the GPU, using the CUDA language. By simply loading the depth-images to the GPU and calling the kernel to compute the gSMOOTH algorithm for each pixel in the image, the processing time already decreases by a factor of 7. Since we perform a lot of memory accesses, especially for data sorting and median calculation, we can benefit from the fast memory access method using shared memory. Thus, we built up a special memory management to increase the speed.

The CUDA kernel is called with one thread per image pixel, which means $Y \times X$ threads for the depth images; here, Y is the width and X is the height of an image. All the CUDA threads within one block load their required image data from the global memory to the shared memory on each time step as shown in Figure 3.14 for a window size of 3. The required sorting algorithms now can be performed on the shared memory, which is much faster than doing it on the global memory. Also, the other calculations profit from the fast memory access, the results are shown in Table 3.1 in Section 3.3.1.

3.3 Results and discussion

We conducted several tests to analyze the efficiency of our proposed depth image enhancement (gSMOOTH) approach. We used various sample depth sequences recorded with Kinect V1 and also common reference depth sequences from Camplani et al. [27], Islam et at. [47] and Middlebury RGB-D database [87]. We recorded the data with a Kinect V1 sensor since most of the reference works also use such devices for acquisition. For gSMOOTH, we used a 5×5 spatial neighborhood of the depth pixels so that the coherency of the recovered depth values are maintained and, we used 10 frames in the temporal window to keep the latency low.

We compare gSMOOTH to six state-of-the-art methods: Avetisyan et al. [44], Camplani et at. [28], Garcia et al. [34], Islam et al. [47], Wang et al. [36] and Nguyen et al. [86]. For establishing an impartial comparison ground for the tests, we set the parameters of each approach to the respective optimum values given in these works. We performed our tests on a Ubuntu PC with Intel i7 3.00 GHz processor, 64GB RAM, and NVIDIA GeForce

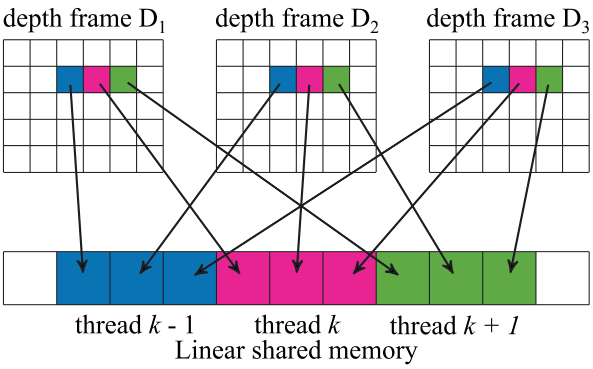


Figure 3.14 – Data management for shared memory.

GTX TITAN X GPU. In the following sections, we present the qualitative and quantitative performance comparison respectively.

3.3.1 Evaluation using real-world depth data

We conducted a qualitative evaluation of our method by using several depth sequences. As we did not come across any benchmark depth sequence which is recorded with stationary RGB-D sensors, we used self-recorded and reference depth sequences from [27] and [47]. We refer the depth sequences on the 1st and 2nd rows of Figure 3.15 as *PersonBox* and *Chairs*; the depth sequences on the 1st, 2nd and 3rd rows of Figure 3.17 as *OfficeRoom*, *PersonWalking* and *PersonSitting*; the depth sequence on the 2nd row of Figure 3.19 as *PersonChair*. The *Chairs* and the *PersonWalking* datasets are from [27] and the *PersonChair* dataset is from [47].

Figure 3.15 shows the performance of gSMOOTH on datasets *PersonBox* and *Chairs*. As we increase the spatial neighborhood from 3 to 5, the results become more visually appealing (see difference between Figure 3.15(b) and (c)). The noise and missing depth information, both in static and dynamic parts of the images, have been significantly reduced by our gSMOOTH. Moreover, we also eliminate the flickering artifacts notably.

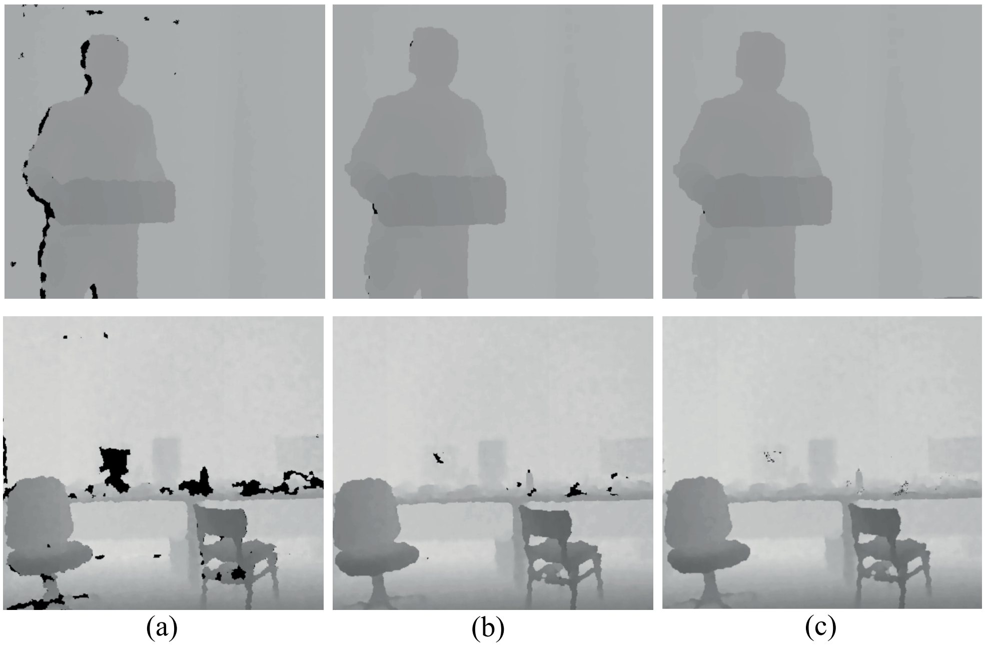


Figure 3.15 – Performance of gSMOOTH on *PersonBox* dataset (1st row) and on *Chairs* dataset (2nd row): (a) raw depth frames, (b) our result with 3 × 3 spatial neighborhood, (c) our result with 5 × 5 spatial neighborhood.

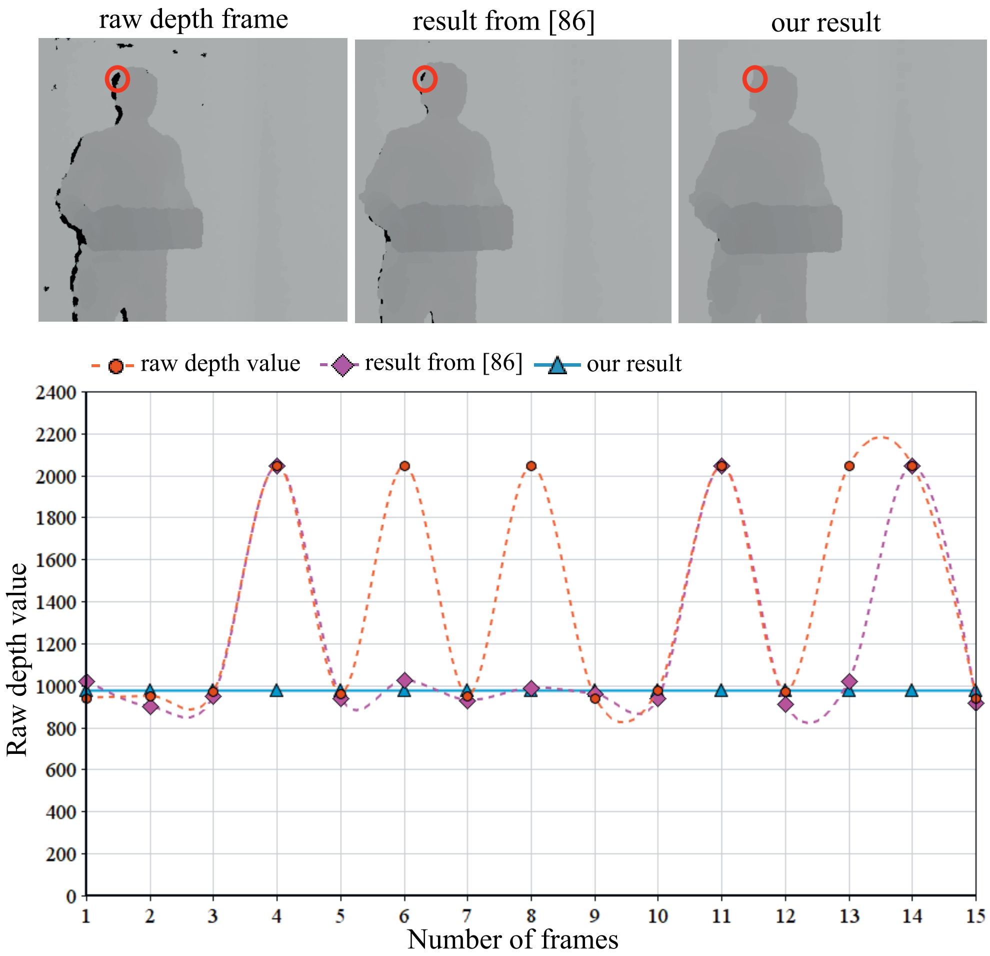


Figure 3.16 – Performance comparison of gSMOOTH and Nguyen et al. [86] on the *PersonBox* dataset for removing noise and flickering: the top left image is a raw depth frame, the middle image is the output from [86] and the right image is from gSMOOTH. The plot shows the performance of gSMOOTH and method [86] over 15 enhanced frames. The depth values on the plot are the recovered depth values for one pixel in the circle-marked area on the depth frames. gSMOOTH significantly reduces the noise and flickering as seen by the blue line on the plot.

Figure 3.16 shows a performance comparison, in noise and flickering artifacts removal, between our method and the approach of Nguyen et al. [86].

Using the reference depth sequence *PersonWalking* and our self-recorded sequence *OfficeRoom*, we can also compare the performance of our gSMOOTH method against reference methods (Avetisyan et al. [44], Garcia et al. [34], Nguyen et al. [86] and Wang et al. [36]) for the static and dynamic scene parts. For both the depth sequences, our method reduces the holes and flickering in the static *and* dynamic parts of the frames, see Figure 3.17 and Figure 3.18 for the results. Here, we can see that the reference methods and our method produce nice results and the surfaces of the objects are nicely recovered. However, we notice that the reference methods suffer along the edges of the objects. The result from method [36] shows blurry edges, whereas our method yields sharp edges; see the zoomed parts in Figure 3.17. Such artifacts are also produced by the approaches from [34] and [44] (see Figure 3.18). The blurry edges typically occur due to the inaccurate registration between the color and depth frames. Since the performance of these reference methods relies heavily on accurate depth and color frame registration, they often suffer from the aforementioned artifacts on both static and dynamic parts of a frame. Since our gSMOOTH method does not require the color frames to enhance the depth frames, it does not exhibit such artifacts. Table 3.1 shows the performance of our method in frames per second *fps* on our test datasets. A separate table for the comparison of performance on CPU among gSMOOTH and other methods is not included here, because we found that the approaches with good quality output required quite a large amount of time, to process one frame, than our gSMOOTH. For example, Wang et al. [36] need 240 seconds to process one frame from the *PersonBox* dataset, whereas our gSMOOTH needs only 0.13 seconds. It is worth to mention that output quality from Wang et al. [36] and gSMOOTH is similar. Since the performance difference in terms of *fps* is quite large, we did not include another table for showing the difference.

Ghosting artifact, on the other hand, is a common problem found with many existing works (e.g., in [44, 47]). We use the reference dataset *PersonChair* from [47] and self-recorded dataset *PersonBox* to demonstrate the performance of our method in removing this artifact. In Figure 3.19(b), (e) and (h), we can observe ghosting artifacts on the rectangle-

Table 3.1 – Performance of gSMOOTH in frames per second (*fps*) on the five test depth sequences.

	CPU	GPU *global* memory	GPU *shared* memory
PersonBox	7.93	55.52	140.25
Chairs	7.53	53.17	135.51
OfficeRoom	8.02	55.81	141.03
PersonWalking	8.48	60.03	144.07
PersonChair	7.76	54.41	137.61

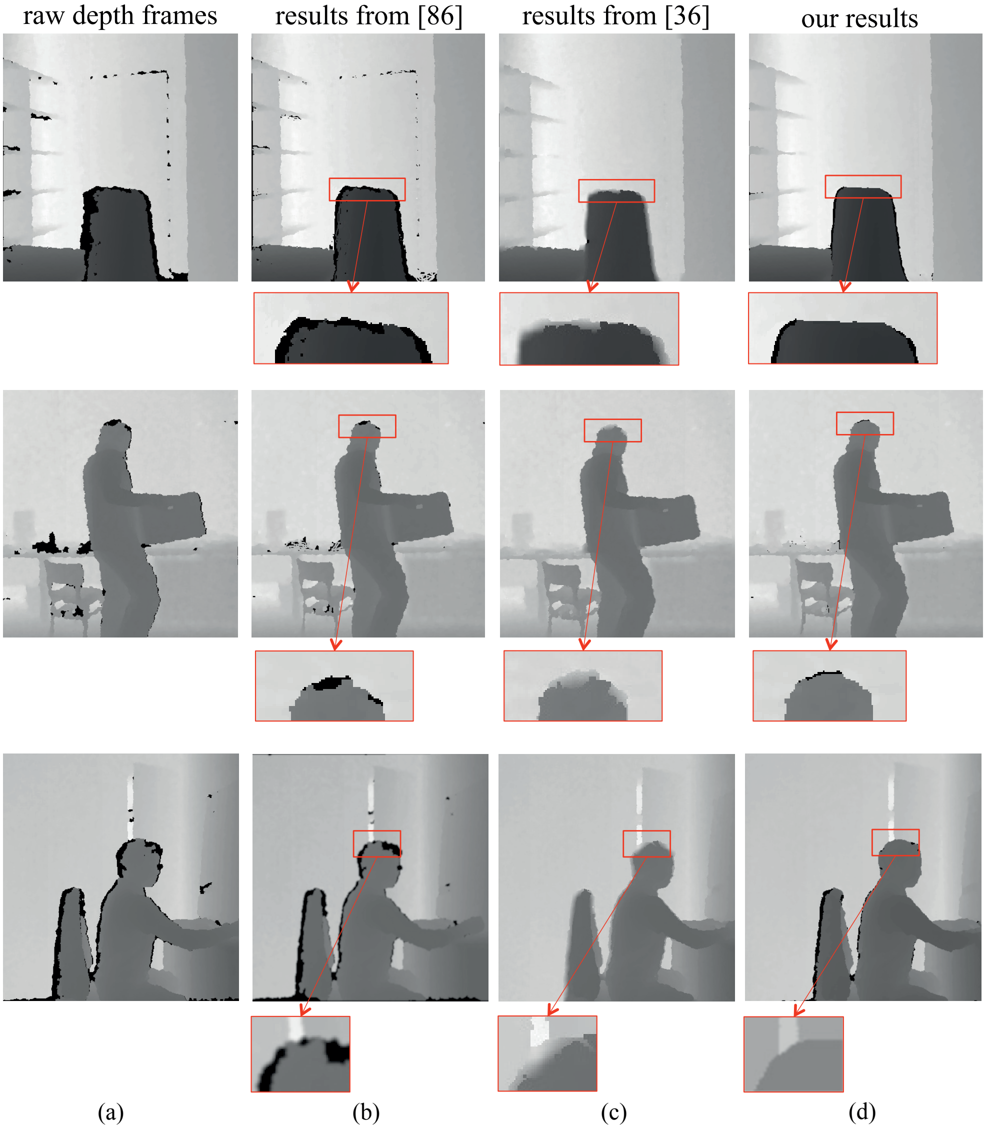


Figure 3.17 – Performance of gSMOOTH on *OfficeRoom* dataset (1st row), *PersonWalking* dataset (2nd row) and *PersonSitting* dataset (3rd row): (a) raw depth frames, (b) results from Nguyen et al. [86], (c) results from Wang et al. [36], (d) our results. For both datasets, our method yields nicer edges while reducing the artifacts.

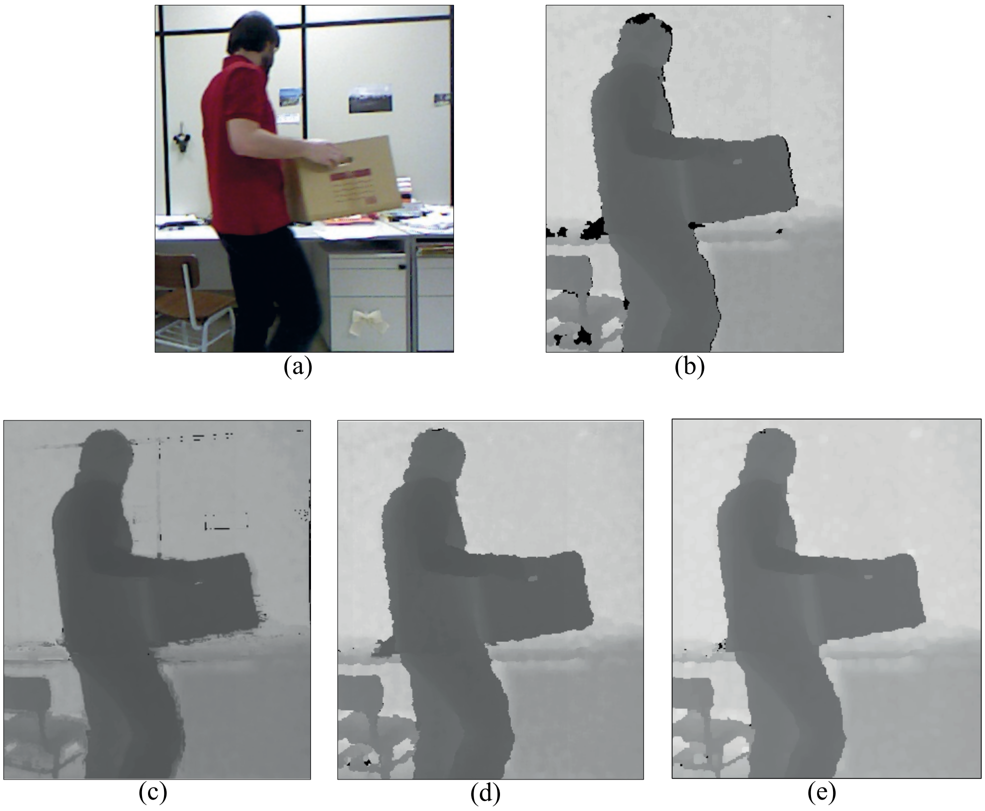
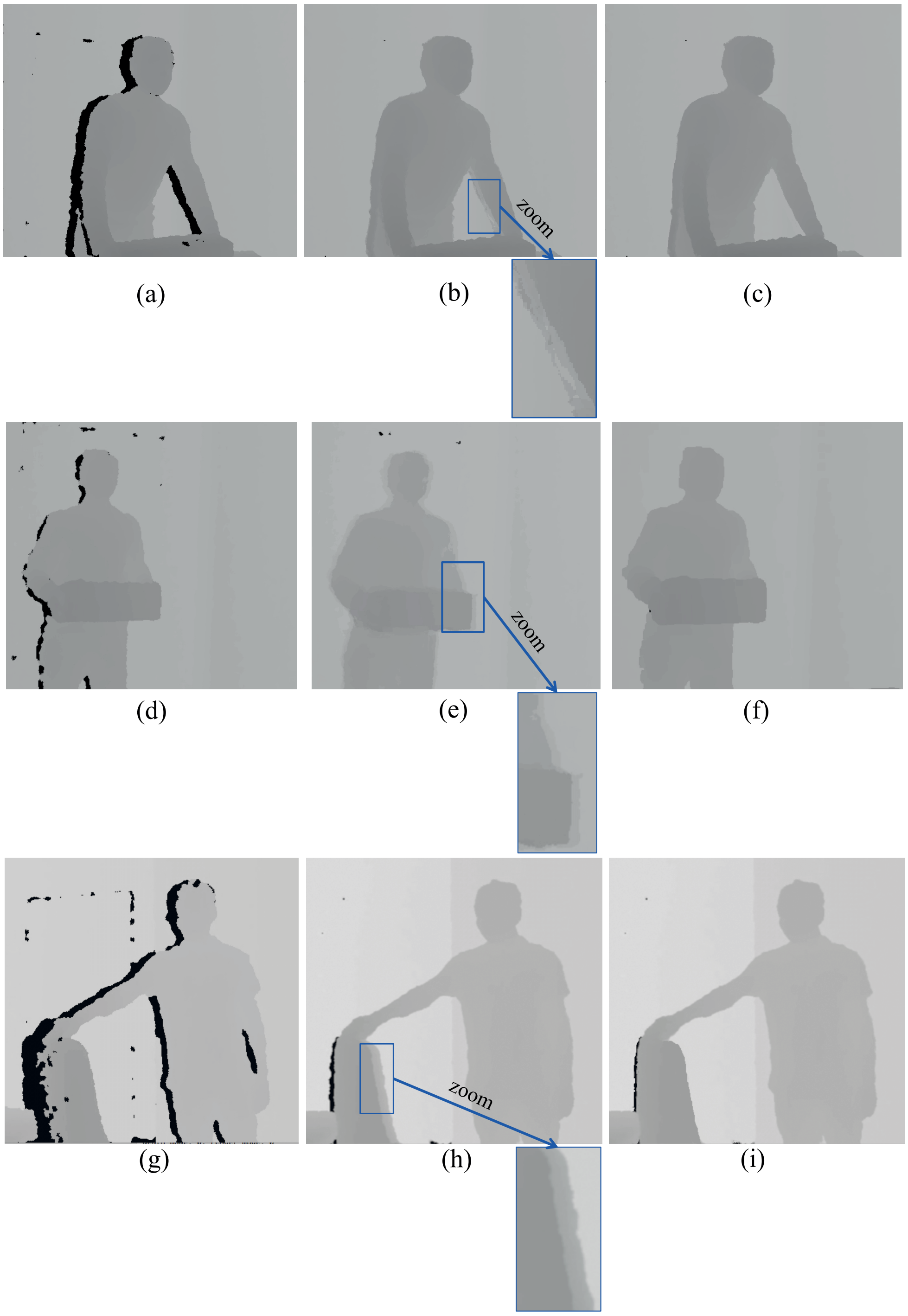


Figure 3.18 – Comparison results-1 on dynamic scenes from *PersonWalking* dataset [27]: (a) color frame, (b) raw depth frame, (c) result from Garcia et al. [34], (d) result from Avetisyan et al. [44], (e) our result. Our result perform well in removing the artifacts and preserving the sharpness of the edges both for static and dynamic parts of the frame.

marked areas due to the person (1st and 2nd rows) and the chair (3rd row) moving rapidly. Figure 3.19(b), (e) and (h) are the result from method [47] and Figure 3.19(c), (f) and (i) show the results from our method. We can see that our method successfully removed the ghosting artifact and reduce the temporal noise while method [47] shows the ghosting artifacts; see the zoomed part in Figure 3.19(b), (e) and (h).

3.3.2 Evaluation using depth data with synthetic degradation

We conducted another test to assess the efficiency of our method using simulated degradation of depth frames. For this test, we took two benchmark depth files (*Book* and *Art*) from the Middlebury depth database [87]. We use the PSNR and SSIM scores to present the results of quantitative assessment. We know that both the ground truth depth information and output (i.e. enhanced) depth information are essential for measuring the PSNR and SSIM scores. Hence, we simulated artifacts similar to those of a Kinect sensor



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.19 – Comparison results-2 on dynamic scenes from *PersonBox* dataset (1st and 2nd rows) and *PersonChair* dataset [47] (3rd row): (a,d,g) raw depth frames, (b,e,h) result from Islam et al. [47] (ghosting artifacts are visible in the rectangle-marked area and in other areas), (c,f,i) our results with gradient-based preprocessing (ghosting artifacts are removed). Our results in (c), (f) and (i) do not exhibit ghosting artifacts since we used the gradient-based spatial preprocessing.

on the benchmarks ground truth depth data by applying the approach of [7]. On these synthetically degraded frames, we apply our method and get the results which are then compared against the ground truth depth frames.

For gSMOOTH, we need successive depth frames in the temporal window; however, the Middlebury database does not deliver such depth sequences. Hence, we created our own sequence of depth frames by copying the benchmark depth images multiple times and applying the method from [7] for yielding randomly distributed artifacts. The result is a static scene where the degradations are randomly distributed over consecutive frames. We then applied our proposed approach to this degraded depth sequence to finally enhance it again. For the reference methods [28, 36] to which we compare the performance of gSMOOTH, we also use the color and depth image pairs from the benchmark datasets along with the degraded images. Figure 3.20 depicts the synthetically degraded depth frames and the respective output frames from [28], [36] and our proposed gSMOOTH.

Using the ground truth depth files and the enhanced results of the different approaches, we measured the PSNR and SSIM scores as shown in Table 3.2. While looking at the PSNR and SSIM scores, we can see that our method yields higher PSNR and SSIM values than

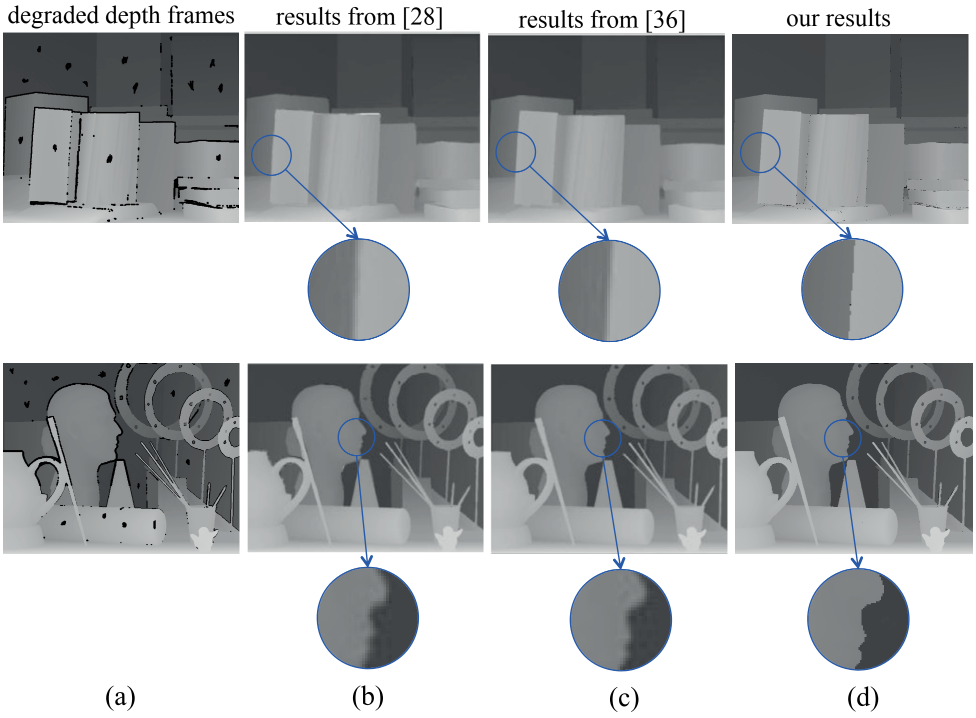


Figure 3.20 – Comparison results on two benchmark depth frames of *Book* and *Art* from Middlebury [87]: (a) depth frames with simulated degradation, (b) results from Camplani et al. [28], (c) results from Wang et al. [36] (d) our results. Our method yields sharper edges, while reducing the artifacts, than the other methods.

Table 3.2 – PSNR and SSIM scores on the results of methods [28, 36] and our gSMOOTH on benchmark depth frames.

	Depth frames			
	Book		Art	
	PSNR	SSIM	PSNR	SSIM
Method [28]	26.5911	0.9416	29.1845	0.9442
Method [36]	26.8854	0.9492	30.3104	0.9587
gSMOOTH	29.7361	0.9735	33.5142	0.9841

the reference methods. Since our method preserves the sharpness of the object's edges and removes most of the artifacts at the same time, it gets a higher score than the reference methods. A careful analysis of the circle-marked areas in Figure 3.20 also denotes the issues with the edges and reveals the reason for the respective SSIM and PSNR scores of the reference methods. All approaches (including ours) recover the depth values of homogeneous surface areas while the reference methods suffer near the edges where the depth discontinuity occurs. Additionally, our method delivers it's output in real-time (at ≈ 140 frames per second, see Table 3.1) due to it's GPU based workflow, whereas the reference methods do not perform in real-time in their original implementation.

3.3.3 Limitations

Although the results of the depth enhancement approach are pleasing qualitatively and quantitatively, it has some limitations. First, we cannot completely reduce the noise that persists in large areas (larger than 5×5 spatial neighborhood) in one region of successive depth frames with our method (see the holes in Figure 3.18(e) and 3.19(f, i). Taking a larger spatial neighborhood of pixels might be useful to refine the large areas with artifacts; however, it can potentially reduce the coherency of the recovered depth value. In this case, the recovered depth might originate from a completely different object that would lead to other artifacts. Thus, we need to understand why and how these artifacts occur. We design a novel depth data capture setup, discussed in Chapter 4, to better understand the depth artifacts and to find a future solution.

Second, our enhanced frames are slightly behind the raw depth data: Abrupt changes in the camera footage need to propagate for a maximum of $n/2$ frames to become finally visible (as seen in Figure 3.10). Thus, in our experiments, we measured a lag of 5 frames at most. Although this is not ideal, we found that i.e., loosing tracked objects, extracting noisy contours, and reconstructing noisy 3D models due to flickering, invalid values, blurry edges, and ghosting artifacts are more severe than this minimal lag.

3.3.4 Future work

Due to the time constraints that cover the scope of this thesis, there is a list of points related to the work that can further be investigated. We address some of the issues in the following.

- **Filling up the large holes.** As we can see that our proposed method removes the ghosting and flickering artifacts completely in real-time, but for some large holes, it cannot fill them completely. We can think of using spatial filters, such as in [38], which use neighbor pixels to fill the holes and in addition, we can think of using structure-aware filters, such as in [88, 89], which would help to maintain the homogeneity of the objects' contours. Including these additional processes into our current workflow might demand extra processing power, however, we can think of optimizing the processing by using hardware-based acceleration methods such as by using GPU coding.

- **Apply our method on other ToF cameras.** Here in this section, we have shown the results that were conducted using a Kinect V1 camera which has a structured light sensor. Although we showed some results of using our method on the depth images from Kinect V2 camera (see Figure 6.4 in Section 6.2) that has a ToF sensor, we did not have enough time to test with other ToF sensors. Since our method considers the invalid values as outliers and then remove them and stabilize them by using the valid depth values from spatio-temporal neighborhood, we can think of applying our approach to remove the 'flying pixel' issues [90] found with the depth images from ToF sensors. Flying pixels occur near the edges of objects where the depth discontinues, i.e. where the near-infrared light emitted by the ToF camera gets reflected in part by an object in the foreground and in part by an object in the background [90]. Here those flying pixels values can be considered as outliers and they can be replaced with valid values by using our proposed spatio-temporal filter.

- **Apply our method on ToF scanners.** Recently we came across some emerging ToF scanners, such as the ibeo LUX [91] and the Eco Scan FX8 [92] that yield low-resolution depth images with better depth accuracy than current ToF cameras [1]. However, since these scanners create these depth images successively point by point, introducing time delay between them, we perceive motion artifacts on those depth images. We can further investigate and adapt our proposed depth enhancement method, which addresses motion artifacts such as ghosting, to attenuate the motion artifacts found with the depth images from these ToF scanners.

Chapter 4

Depth noise extraction and visualization

This chapter presents our novel depth data capture setup that we design and develop to understand the nature and reason of depth noise that a depth sensor yields. We analyze the experimental data acquired with this setup and show how the distortion from a depth camera is related to the distance of the objects from the camera, the viewing angle of the camera and the lighting condition of the scene. Hence, our experiments have three parameters – the distance of the objects from the camera, the viewing angle and the lighting condition. With the data obtained from this setup, we generate ground truth data which we use to extract noise from our captured test data (which typically contains noise) to visualize the noise or distortion produced by a depth camera for those three parameters. In the following sections, we discuss the motivation behind the work, the process of generating ground truth data using markers on the depth camera and a target object and the process of extracting the noise and visualizing it.

4.1 Problem description and motivation

While testing our proposed depth enhancement strategy on our test data sets, we noticed that the number of required frames in the sliding window varies depending on the distance of the object from the location of the depth cameras, the viewing angle of the camera and the lighting condition. We observed that for different distances, the noise or the error generated on the depth frames vary and hence we needed to use different number of frames in the sliding window for different distances to remove this noise. Since our test data sets were recorded in different lighting conditions (e.g. natural daylight, room light, studio light), we also observed different error characteristics for these different lighting conditions and we also noticed variations in the quantity of depth errors for different viewing angles. Hence, here we investigate the characteristics of the depth noise and, quantify and visualize the noise characteristics to address these following three queries:

- how the variations of distances of the objects from the depth camera affect the quantity of noise
- how the different lighting conditions influence the quantity of noise on the depth frames and
- how the noise changes with the change in the viewing angle of the camera.

Once we would be able to visualize and analyze the relationship of the noise characteristics with respect to the distance of objects from camera, lighting condition, and viewing angle, we would better understand the nature of noise in different situations and this could essentially help us to potentially optimize our proposed depth enhancement strategy and could also lead us towards a future new depth image enhancement solution.

4.2 Noise extraction and analysis method

In order to visualize and analyze the noise characteristics of a depth sensor, we design an experiment where we generate ground truth data for a target object. Here, we register a depth camera in the global coordinate system of a tracking system that is also used to continuously deliver global coordinates of the target object's surface that is visible by the camera. Using the common coordinate system, the synchronization between the devices, and the known surface geometry, we could eventually be able to reconstruct the projection area and the undistorted depth of the surface in every camera image. By simply subtracting this ground truth from recorded depth data, we can obtain the plain noise in a sequence of depth images. To make our experiments most effective, we automate the test sequences. We vary the lighting condition over time and the target object's distance is varied from the depth camera by moving the target on a movable platform that also provides support for the camera and the target object to be attached steadily. Besides this, the used target's surface is arched in order to cover the dependence on the viewing angle in every shot.

For the visualization and analysis of the noise, we develop software that helps to extract the noise from the test sequences and encodes the absolute noise as colored textures mapped to the ground truth planes. We generate such error mapping on ground truth data for each of the three conditions stated in Section 4.1.

4.2.1 Sources of depth noise

While capturing a scene with consumer depth cameras, such as Microsoft Kinect, we perceive different artifacts on the captured depth images. Whether the depth cameras use a structured light pattern or time-of-flight ToF method to measure the depth of the objects inside the captured scene, there usually are sources of errors or noise that cause invalid or inaccurate measurement of the depth of the objects. In the case of structured light devices, the projected pattern might become too weak compared to the background light that causes the erroneous depth generation. The typical reasons for the weakening of the projected pattern cover a wide range of issues such as:

- ambient light [1],
- external active illumination source interference [2],
- active light path error caused by reflective surfaces,
- occlusion,
- erroneous light pattern detection in dynamic scenes,
- depth offset for non-reflective objects and others [2].

ToF cameras also produce depth error that mainly originate from the electronic noise, dark noise and photon shot noise of the camera sensor [12]. Electronic noise is a random fluctuation that is characteristic of all electronic circuits such as analog to digital converters. Dark noise summarizes additional photodetector noise sources such as thermal noise, i.e., random fluctuations due to changes in temperature. Photon shot noise occurs due to the photon character of light [1]. These sources of noise are caused due to the following reasons:

- background illumination – since most ToF sensors use infrared light, the unfiltered external background light interferes with the depth calculation. This problem becomes more severe in capturing outdoor scenes.
- reflections – surfaces reflecting the infrared light might cause the erroneous depth measurements. For example, specular or highly reflective surfaces cause the superimposition of measured values. While capturing the scene for generating ground truth data, we noticed that even diffuse surfaces turn into specular surfaces when the viewing angle is changed which cause unwanted reflections.
- temperature – when some ToF sensors, e.g., Microsoft Kinect V2, becomes warm due to higher consumption of energy, this increase in device temperature can potentially lead towards a shift in distance values during this increased temperature phase [1].

Therefore, we set three parameters for our experiment where we use varying lighting conditions, varying viewing angle and varying distance to quantify and visualize the error on the captured object. We consider the variation in distance since the depth sensors generate different quantity of error based on the distance of an object from the camera [27, 86].

4.2.2 Experimental setup

For our experiment, we used a moveable target and a fixed depth camera; the camera is able to rotate in vertical and horizontal directions. We mounted the test target and the depth camera on a steady structure made from aluminium. The steady structure helps to maintain a stable position of the target and the depth camera while recording the position of the camera and the target, in the world coordinate system, for various distances of the target object from the depth camera.

4.2.2.1 Description of the apparatus used in the experiment

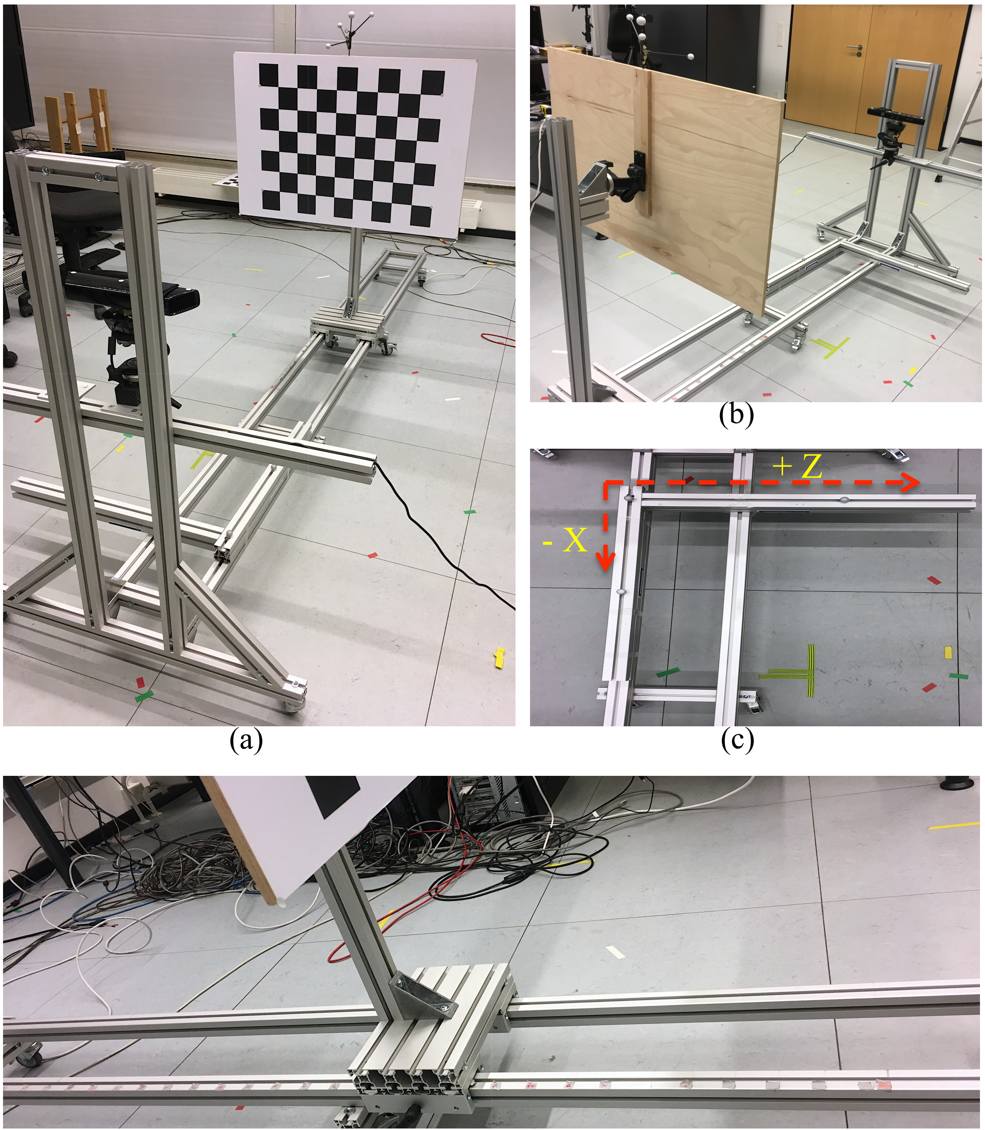
The steady structure

For conducting the experiments, we mounted the target checkerboard and the depth camera on an aluminium-made steady structure which we built in order to keep the position of the target object and the depth cameras as steady as possible in the world coordinate system of the tracking system. Figure 4.1 shows the steady structure along with other apparatus we used in our experiment. The length of the steady structure is 3.2 meters and the height is 1.2 meters from the ground. There are locking mechanism on each of the eight wheels that keeps the position of the whole structure stable. There are two long rails on the bottom of the structure to support the steady movement of the test target. We attach, at one end of the structure, a depth camera and on the other end, a test target.

The test target is attached to a moveable platform that can be moved along the two-rig rail system. The height of the rigs, on which the depth camera and the test target are attached, are adjustable and the test target can be moved in both directions, forward to backward and vice versa. It is worth to mention that, we also placed the 'L-shaped' ground plane, required by the OptiTrack tracking system [93] on the bottom rails so that its position also remains stable.

The target object

We use a checkerboard as our target object that has a planar surface. We have attached the checkerboard on a wooden pressboard and carefully glued it on the wooden board so that the printed checkerboard is attached properly on all areas of the wooden board; this ensures that the surface of the target is planar and all the pixels of the target are located on the same plane. This reduces the possibility of the occurrence of external errors that might be caused due to an uneven surface of the target object. We avoided using boards constructed from lighter material such as cardboards that would not be able to maintain their original structure and might bend at certain places after a while. Moreover, the flexible nature of these lighter boards was also not a good fit for our experiment since it would induce measurement errors. We used a matte print of the checkerboard to keep



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

Figure 4.1 – Images of our experimental setup for the steady structure: (a) steady structure with depth camera and test target – view-1, (b) steady structure with depth camera and test target – view-2, (c) 'L-shaped' ground plane, required by OptiTrack tracking system for getting the world coordinate system, (d) steady structure – the two-rig rail system on which the target moves to the marked positions.

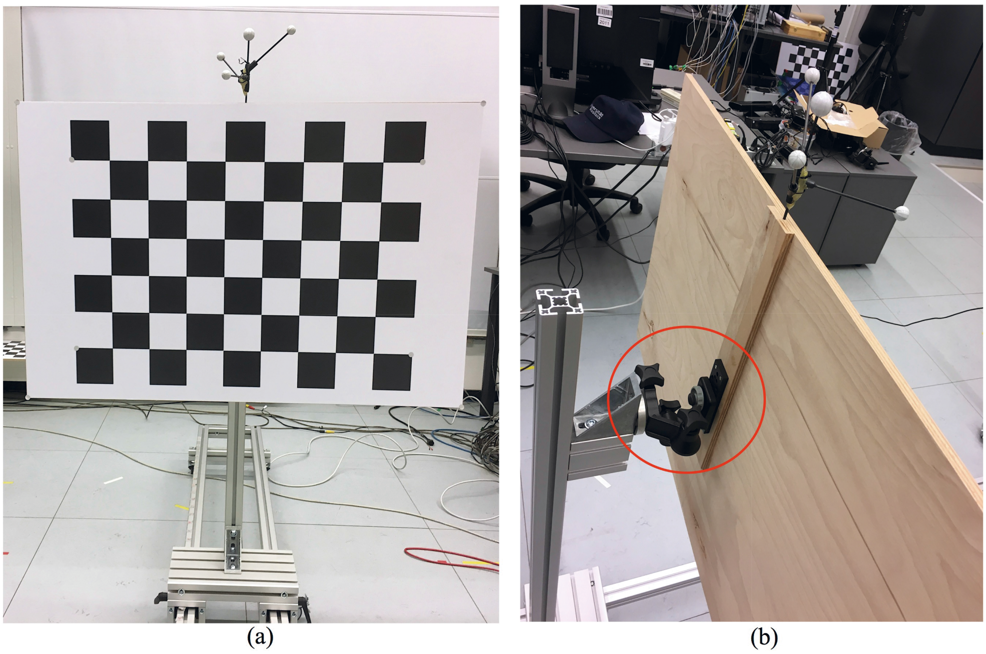

(a) (b)

Figure 4.2 – Images of our experimental setup for the test target: (a) the test target attached on the moveable platform on the steady structure, (b) the test target mounted to the steady structure with the camera gear.

the errors occurred due to reflection at a minimal level. The height and width of our target checkerboard is 0.57 × 0.85 meters. We mounted the checkerboard target to the steady structure using a mounting gear usually used to mount digital cameras to tripods. We adjusted the position of the target so that it levels up with the ground plane of the tracking system (details of the tracking system is discussed later in this section) and it is perpendicular to the recording depth camera.

For extracting the noise characteristics that occur due to the variations of the distance of the target object from the camera, we moved the target object with the range of distance from 1.2 meters to 2.6 meters, from the depth camera, with an interval of 0.05 meters. Although this range of distance does not accurately cover the original minimum and maximum operating distance of a Kinect V1 sensor (which can measure the depth of objects locating in the range between 0.5 meters to 4.5 meters from the camera sensor) or a Kinect V2 sensor (the maximum depth-sensing distance is 8 meters), since the generated noise at the extreme ranges was too high to extract any usable information, we used the distance range from 1.2 meters to 2.6 meters. Moreover, for our setup, if we placed the target closer than 1.2 meters from the camera, the target was beginning to crop near the edge of the object on the recorded depth image while rotating the camera for varying viewing angles. We used a measurement ruler to mark the different distances of the target object from the depth camera. We also marked the intervals with the ruler on the steady structure (see the marks on the two-rig rail system in Figure 4.1(d)) with permanent marker to avoid any measurement errors. This distance marking helped us to make precise (as precise as manually possible) movement of the target object on the steady structure in either direction from the depth camera. Figure 4.2 shows the target we used in this experiment.

The recording depth camera

We used both the Kinect V1 and V2 depth cameras to record the target object for our experiment. Finally, we used the output from Kinect V2 to show the different results in the chapter, since this is the most recent product of Microsoft Kinect and it has better depth resolution than Kinect V1. However, in the results section, we used the depth images from Kinect V1 and Kinect V2 to compare and explain certain noise characteristics in their respective images. We mounted the depth camera on the aluminium-made steady structure to ensure its stable position. To rotate the camera to various angles, we attached it to a rotatable gear similar to the ones used in digital photography. For our experiment, we considered only the vertical rotation of the camera by which we get different viewing angles with respect to the movable target. We also attached the power cable of the camera to the steady structure so that there is no unwanted movement of the camera occurred due to the movement of the power cable. Similar to the test target, we adjusted the location and orientation of the depth camera so that it can level with the ground plane of the tracking system and it is perpendicular to the test target. This adjustment is done to minimize the external errors caused due to improper alignment of test equipment. Figure 4.3(c) and (d) show the depth cameras, with tracking markers on them, used in this experiment.

The tracking system and placement of the markers

For obtaining the position of the depth camera and the target while varying the distance of the target from the camera, we used 12-DOF (degree-of-freedom) OptiTrack tracking system [93]. We used 12 tracking cameras that track the position of the target and the camera by locating and tracking the markers placed on the test target and the depth camera. We discussed placing the ground plane for obtaining the world coordinate system for the tracking system earlier in this section and in Figure 4.1(c). The tracking cameras are placed near the ceiling on another four-sided steady aluminium-made structure which surrounds the total area covered by our hardware setup. We placed three cameras on each side of the four-sided structure so that they can cover all possible movements of the target inside the experiment area. Figure 4.4 shows an illustration of the placement of the 12 tracking camera on the four-sided rig, and the placement of the depth camera, test target and the steady structure which we used in this experiment.

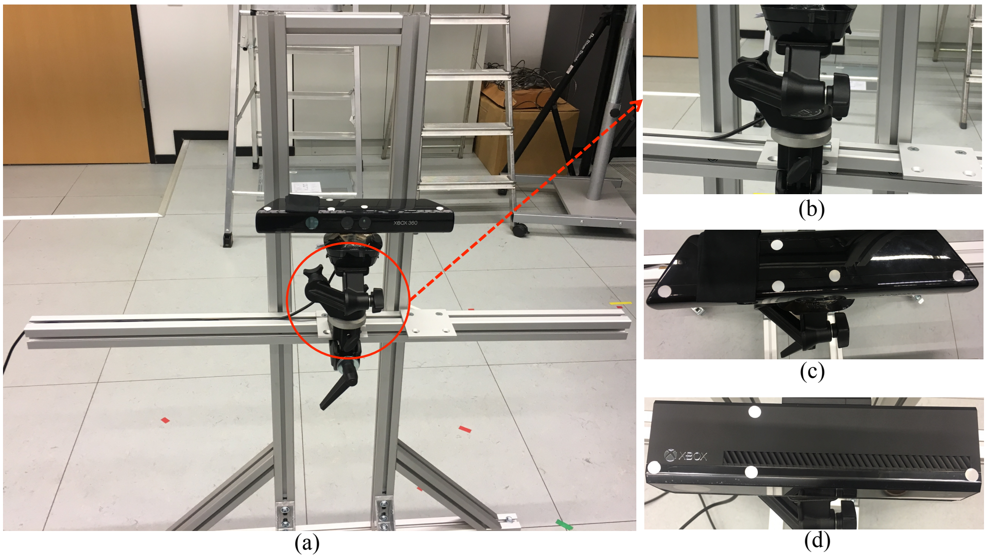


Figure 4.3 – Images of our experimental setup for the depth camera: (a) the depth camera attached to the height-adjustable rig of the steady structure, (b) the depth camera mounted to the steady structure with the camera gear, (c,d) markers on the Kinect V1 and V2 depth cameras.

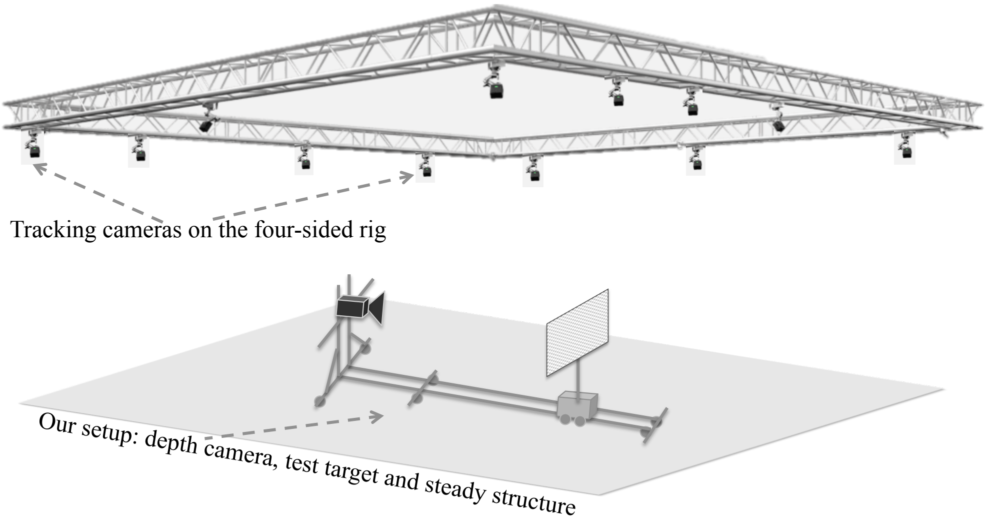
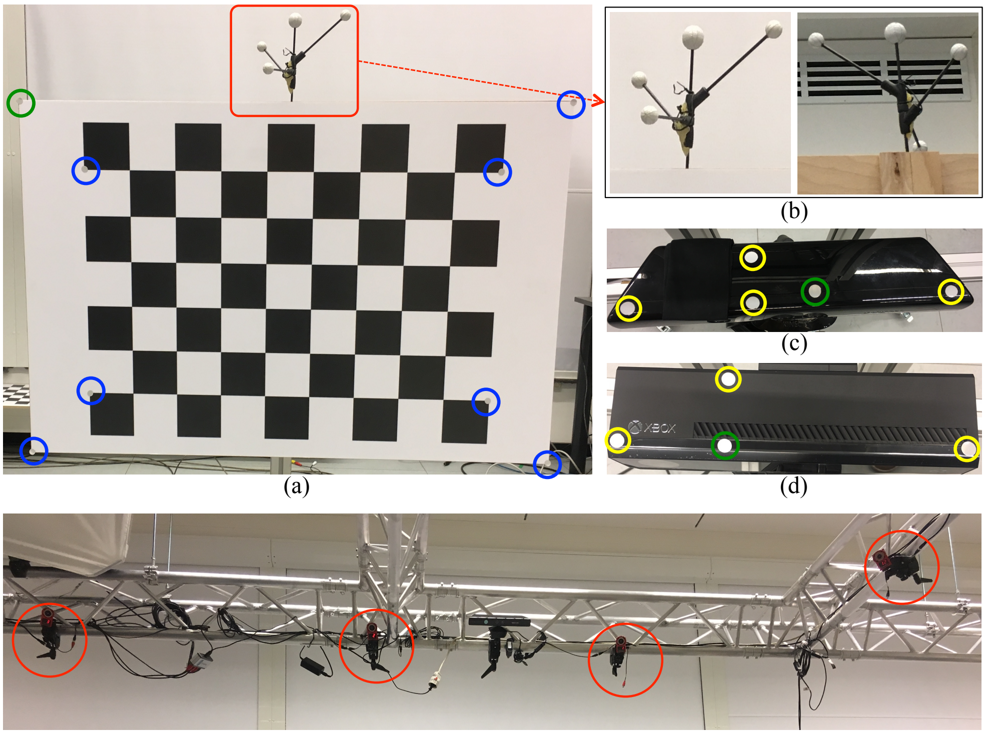

Tracking cameras on the four-sided rig
Our setup: depth camera, test target and steady structure

Figure 4.4 – Illustration of our tracking setup along with placement of the 12 tracking cameras, recording depth camera, test target and the steady structure.

In order to track the position of the depth camera and the test target, we placed markers on both the depth camera and the checkerboard, see Figure 4.5. We placed several markers along the length and width of both Kinect V1 and V2. The green circle marked marker, in Figure 4.5(c), is placed directly on top of the depth sensor; this way we



(a)

(b)

(c)

(d)

(e)

Figure 4.5 – Placement of markers on the depth camera and the test target and the position of part of the tracking cameras. Here, in (a) we can see that the test target (the checkerboard) is equipped with a asymmetric structure and another eight markers which provides a stable and steady position in the world coordinate system. Since the eight markers are on the same plane, we needed to attach the asymmetric structure to create a trackable rigid body for the OptiTrack system. A zoomed in version of the asymmetric structure in (b) shows both the front and behind view of the structure. The green marked marker is considered as the pivot of the rigid body. The depth cameras in (c) and (d) are equipped with several markers, along their length and width directions, which also helps to create a trackable rigid body for the OptiTrack system. Here the green marked markers, in (c) and (d) are placed on top of the estimated position of the depth sensor. We used this marker to create the pivot for this trackable rigid body. Finally, a part of the placement of the tracking cameras on the four-sided rig attached near the celling of the room are shown in (d).

get the position of the depth sensor in the world coordinate system. However, we needed to make some adjustments to get the real position of the depth sensor (the actual depth sensor is located little below the top surface of the depth camera) that we will discuss later in this chapter. We placed the other markers (placed along the length and the width of the depth camera) on the depth cameras so that the rigid body we created on the OptiTrack system can be stable and provide the position values in X, Y and Z coordinates.

There are eight markers on the surface of the test target which we used along with an asymmetric structure attached on top of the test target to create a rigid body for the test target on the OptiTrack system, see Figure 4.5(a). Since the eight markers are located on the same surface, we attached this asymmetric structure so that the rigid body can provide the position of the four corners of the test target that we later use to create the ground truth image. We consider the upper-left marker (marked with green circle) as the pivot of this rigid body and calculate the position of other four corners using the dimension values of the checkerboard. Figure 4.5(e) shows parts of the placement of the tracking cameras on the rig attached to the ceiling.

Software tools and libraries used to generate the ground truth data

We developed software to generate ground truth data from the tracked position of the depth camera and the test target. The software gets the position of the two rigid bodies – one for the depth camera (we named it 'Kinect') and another for test target (we named it 'Checkerboard'). We used the VRPN streaming tool [94] to get the position of the rigid bodies from the server machine (which records the positions of the rigid bodies) to the client machine (where we perform necessary calculations to generate ground truth using the tracked data received from the server machine). We used the **Motive 2.0** software platform from OptiTrack to visualize the rigid bodies tracked by the OptiTrack tracking system. Figure 4.6 shows the rigid bodies, the location of the tracking cameras, the placement of the 'L-shaped' ground plane and the direction of world coordinate system for the OptiTrack system. From 4.6(b), we can see that the Z-axis of the rigid body of the depth camera is aligned with the world coordinate system of the OptiTrack system. However, since the test target is in front of the depth camera, the Z-axis should face towards the test target. Therefore, we needed to calibrate the depth camera so that we can obtain its position, being the Z-axis facing toward the test target, in the world coordinate system. The calibration procedure involves translation and rotation steps that we will discuss in the next section. Before performing this calibration, we also needed to adjust the position of both the rigid bodies in their initial coordinate system (i.e. in the world coordinate system) by using the orientation values provided by the OptiTrack system. We needed this step to account for the minor misalignment from the ground plane that the rigid bodies normally have. For quick prototyping, we initially used Matlab and then,

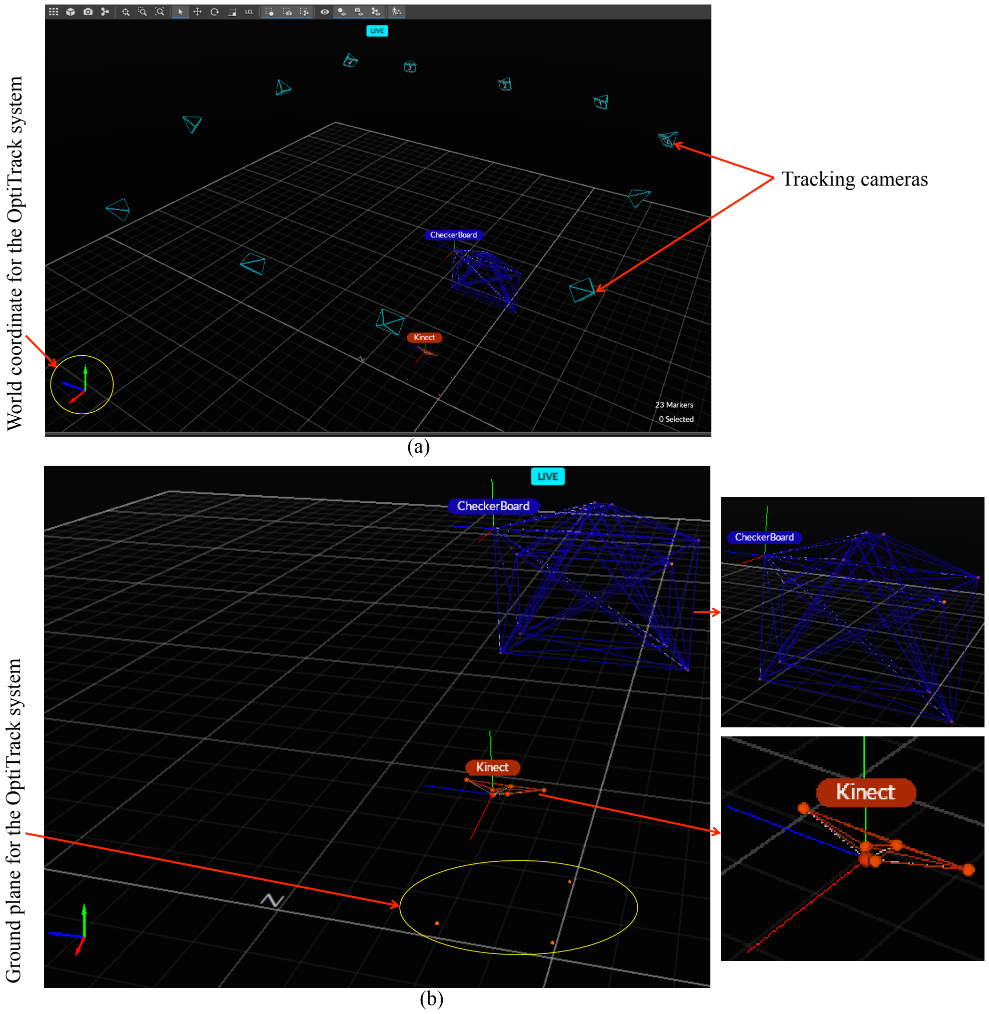


Figure 4.6 – Images from the Motive software of the OptiTrack tracking system showing the placement of the tracking cameras in (a) and the world coordinate system (the red, green and blue arrows represents X,Y and Z axes respectively) is shown inside the yellow marked circle in (a). We can see the rigid bodies (Kinect and CheckerBoard) in (b) which correspond to the depth camera and the test target respectively. In (b), we can see that the coordinate systems of both the rigid bodies align with the world coordinate system provided by the OptiTrack system. The yellow marked ellipse in (b) shows the location of the 'L-shaped' ground plane required by the OptiTRack system to define the world coordinate of the tracking system.

we used C/C++, OpenCV, OpenGL and Matlab to create our software to visualize the recorded depth data by the depth camera, the ground truth data and the noise.

4.2.3 Camera and test target calibration, and ground truth data creation

Since the test target and the depth camera are both positioned in the world coordinate system, we need to transform the test target position to the position of the depth sensor for obtaining the ground truth image. By looking at the local coordinate system of the depth sensor and the test target, in Figure 4.6(b), we find out that both these objects' local coordinates are aligned with the world coordinate system from the OptiTrack system. Hence, we need to perform firstly, a translation process for the test target so that its position is translated to the position of the depth camera and then, we need to perform a rotation operation on this translated position so that the Z-axis of the depth camera, which indicates the distance (i.e. depth) of the objects from the depth sensor, would face towards the test target. We can see, in Figure 4.6(b), that the Z-axis of the depth camera is facing in opposite direction from the test target, see the red line (perpendicular to the blue and green lines) on the 'Kinect' rigid body, which indicates the Z-axis of its local coordinate system.

4.2.3.1 Calibration of the depth camera and the test target

To create ground truth image of the test target using the position and orientation information from the OptiTrack system, we need to calibrate the depth camera so that we get the position of the test target from the perspective of the depth camera's position. Calibration involves two steps that are the translation and the rotation. Using these two steps, we can essentially transform the test target's position to the camera's coordinate system. However, we also need to consider the orientation of the depth camera and the test target to adjust their respective position in the world coordinate system before we perform the translation and rotation steps.

Adjusting the position of the depth camera and the test target using orientation information

Suppose, we have the position (C_p and T_p) and orientation (C_o and T_o) information, see Equation 4.1, of the depth camera and the test target respectively and we would like to perform first an adjustment to the position of the depth camera and the test target using the orientation information and then apply the translation and the rotation steps on the adjusted position of the test target. It is worth to mention that both the position and orientation information for the depth camera and the test target are provided for the pivots of their respective rigid bodies (see the description of the markers in section 4.2.2.1

and Figure 4.5). For adjusting the position of the depth camera and the test target, we calculate the rotation matrix using the orientation information of the camera and the target. We use Equation 4.2 to convert the orientation information to rotation matrices R_c and R_t; R_c and R_t refer to rotation matrices for the depth camera and the test target respectively.

$$\begin{aligned} Camera\_position, C_p &= [C_x, C_y, C_z] \\ Camera\_orientation, C_o &= [C_{o_x}, C_{o_y}, C_{o_z}, C_{o_w}] \\ Target\_position, T_p &= [T_x, T_y, T_z] \\ Target\_orientation, T_o &= [T_{o_x}, T_{o_y}, T_{o_z}, T_{o_w}] \end{aligned} \tag{4.1}$$

$$\begin{aligned} R_c &= \begin{pmatrix} 1-2(C_{o_y}^2+C_{o_z}^2) & 2(C_{o_x}C_{o_y}+C_{o_w}C_{o_z}) & 2(C_{o_x}C_{o_z}+C_{o_w}C_{o_y}) \\ 2(C_{o_x}C_{o_y}+C_{o_w}C_{o_z}) & 1-2(C_{o_x}^2+C_{o_z}^2) & 2(C_{o_y}C_{o_z}+C_{o_w}C_{o_x}) \\ 2(C_{o_x}C_{o_z}+C_{o_w}C_{o_y}) & 2(C_{o_y}C_{o_z}+C_{o_w}C_{o_x}) & 1-2(C_{o_x}^2+C_{o_y}^2) \end{pmatrix} \\ R_t &= \begin{pmatrix} 1-2(T_{o_y}^2+T_{o_z}^2) & 2(T_{o_x}T_{o_y}+T_{o_w}T_{o_z}) & 2(T_{o_x}T_{o_z}+T_{o_w}T_{o_y}) \\ 2(T_{o_x}T_{o_y}+T_{o_w}T_{o_z}) & 1-2(T_{o_x}^2+T_{o_z}^2) & 2(T_{o_y}T_{o_z}+T_{o_w}T_{o_x}) \\ 2(T_{o_x}T_{o_z}+T_{o_w}T_{o_y}) & 2(T_{o_y}T_{o_z}+T_{o_w}T_{o_x}) & 1-2(T_{o_x}^2+T_{o_y}^2) \end{pmatrix} \end{aligned} \tag{4.2}$$

We obtain the adjusted positions of the depth camera and the test target using the formula in Equation 4.3. Since we placed four markers on the four corners of the test target and considered the upper-left marker as the pivot of the rigid body of the test target (see the description of the markers in section 4.2.2.1 and in Figure 4.5), we need to calculate the position and adjusted position of the other three corner markers to obtain the final ground truth plane of the test target. We use the dimension (height and width, see the description in Section 4.2.2.1) information of the test target to calculate the position of the other three corners of the test target. We call the position of the pivot as T_p (already stated in Equation 4.1), the position of the marker on the upper-right corner as T_{p_2}, the position of the marker on the lower-left corner as T_{p_3} and the position of the marker on the lower-right corner as T_{p_4}. The calculation process of the positions of these three corners is stated in Equation 4.4. Then we obtain these corners' adjusted position using the orientation (R_t) of the pivot T_p as stated in Equation 4.5. After adjusting all the corners of the test target, we put these corners in a matrix and call it target adjusted T_a as stated in Equation 4.6. For further calculation, e.g. for the translation and rotation steps, we use this matrix T_a as the position of the test target. It is worth to mention that since we later plotted the four corners in OpenGL to visualize the ground truth plane, we placed nT_{p_4} before nT_{p_3} in T_a, see in Equation 4.6.

$$\begin{aligned} NewCamera\_position, nC_p &= C_p \times R_c \\ NewTarget\_position, nT_p &= T_p \times R_t \end{aligned} \tag{4.3}$$

$$\begin{aligned} T_{p_2} &= [T_p(x - width), T_p y, T_p z] \\ T_{p_3} &= [T_p x, T_p(y - height), T_p z] \\ T_{p_4} &= [T_p(x - width), T_p y, T_p z] \end{aligned} \tag{4.4}$$

$$\begin{aligned} nT_{p_2} &= T_{p_2} \times R_t \\ nT_{p_3} &= T_{p_3} \times R_t \\ nT_{p_4} &= T_{p_4} \times R_t \end{aligned} \tag{4.5}$$

$$T_a = \begin{pmatrix} nT_p \\ nT_{p_2} \\ nT_{p_4} \\ nT_{p_3} \end{pmatrix} \tag{4.6}$$

Translation of the test target position to the depth camera position

Since we need to perceive the position of the test target from the position of the depth camera, we need to perform a translation step on the matrix T_a that contains the four corners of the test target. Translation steps involves subtracting the adjusted position of the depth camera nC_p from the test target corner matrix T_a. Since T_a has four rows in which each row contains three values (the X, Y and Z coordinate values of all the four corners) and nC_p has just one column with three values (the X, Y and Z coordinate values of the marker placed on top of the depth sensor location), we need to create a matrix with four columns for the depth camera where each column would contain the X,Y and Z coordinate values of the depth sensor. The camera matrix C_a, in Equation 4.7 shows the elements of this matrix. Since we need to subtract the same camera position from each of the corners, we fill the camera matrix C_a where all the rows have the same X, Y, and Z coordinate values. For performing the translation, we perform a subtraction and obtain the translated target corners trT_a as stated in Equation 4.8. On this translated target corners trT_a, we perform a rotation operation that is stated in the next section.

$$C_a = \begin{pmatrix} nC_p \\ nC_p \\ nC_p \\ nC_p \end{pmatrix} \tag{4.7}$$

$$\begin{aligned} trT_a &= T_a - C_a \\ &= \begin{pmatrix} nT_p \\ nT_{p_2} \\ nT_{p_4} \\ nT_{p_3} \end{pmatrix} - \begin{pmatrix} nC_p \\ nC_p \\ nC_p \\ nC_p \end{pmatrix} \end{aligned} \tag{4.8}$$

Rotation of the test target position

From the direction of the X, Y, and Z coordinates of the depth camera (shown in Figure 4.5 of section 4.2.2.1), we can see that the viewing direction of the depth camera is in the opposite direction of the test target. Hence, we need to perform a rotation so that the Z-axis of the camera face the test target. To do so, we rotate the X-coordinate of the translated target corners trT_a to -90 degrees and later rearrange the coordinates so that they match with the coordinate system of OpenGL platform. We rotated the X-coordinate in the opposite direction, because this X-coordinate is basically giving us the distance of the test target from the depth camera. Hence, we consider the X-axis values as the Z-axis values and vice versa. For this rotation, we calculate the rotation matrix R_x as stated in Equation 4.9 which we multiply with the translated target corners trT_a, see Equation 4.10 and finally we get the final position of the test target corners T_{fin} which we use to plot the ground truth plane of the test target.

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \tag{4.9}$$

$$T_{fin} = trT_a \times R_x \tag{4.10}$$

T_{fin} contains the X, Y, and Z coordinate values for all the four corners of the test target after it has been translated to the position of the depth camera and then rotated -90 degrees in X-coordinate. Equation 4.11 shows the contents of $T_f in$; here T_{c_1},T_{c_2},T_{c_3} and T_{c_4} are the final transformed the upper-left, upper-right, lower-left and lower-right corners of the test target respectively. We swap the columns to rearrange them so that the first, second and third columns contain the X, Y, and Z coordinate values respectively, see Equation 4.12. For plotting the ground truth plane, using these four corner positions, in OpenGL, we use Equation 4.13; the process of creating the ground truth in OpenGL is explained in Section 4.2.3.2.

$$T_{fin} = \begin{pmatrix} T_{c_1}(Z) & T_{c_1}(X) & T_{c_1}(Y) \\ T_{c_2}(Z) & T_{c_2}(X) & T_{c_2}(Y) \\ T_{c_3}(Z) & T_{c_3}(X) & T_{c_3}(Y) \\ T_{c_4}(Z) & T_{c_4}(X) & T_{c_4}(Y) \end{pmatrix} \tag{4.11}$$

$$T_{fin} = \begin{pmatrix} T_{c_1}(X) & T_{c_1}(Y) & T_{c_1}(Z) \\ T_{c_2}(X) & T_{c_2}(Y) & T_{c_2}(Z) \\ T_{c_3}(X) & T_{c_3}(Y) & T_{c_3}(Z) \\ T_{c_4}(X) & T_{c_4}(Y) & T_{c_4}(Z) \end{pmatrix} \tag{4.12}$$

$$T_{fin} = \begin{pmatrix} -T_{c_1}(X) & T_{c_1}(Y) & -T_{c_1}(Z) \\ T_{c_2}(X) & T_{c_2}(Y) & -T_{c_2}(Z) \\ T_{c_3}(X) & -T_{c_3}(Y) & -T_{c_3}(Z) \\ -T_{c_4}(X) & -T_{c_4}(Y) & -T_{c_4}(Z) \end{pmatrix} \tag{4.13}$$

Additional adjustments

Using the values from T_{fin}, when we created the ground truth image (discussed in Section 4.2.3.2) and compared against a fitted plane through the recorded depth image (discussed in Section 4.2.5), we found out that there are slight misalignment along the X, Y, and Z coordinates of the test target with the ground truth image, hence, we needed to adjust the angles slightly for the X, Y, and Z coordinates. This misalignment occurs since we placed the markers on top of the depth sensor without knowing exactly at which location the depth sensor is located physically inside the depth camera. Moreover, we also don't know the real physical orientation of the image plane of the Kinect. That's the reason we need to adjust the position and rotation angles of the pivot of the depth sensor. It is worth to mention that we used three horizontal angle position of the depth camera with respect to the test target. For the first position, the depth camera is kept perpendicular to the test target, i.e. the horizontal angle between them is 0. For the second and the third horizontal positions, we rotated the camera by 4° in the right direction and 5° in the left direction respectively. We wanted to rotate the camera by 5° to the right side too, however the tracking cameras were not able to track the markers beyond 4° for that position of the depth camera.

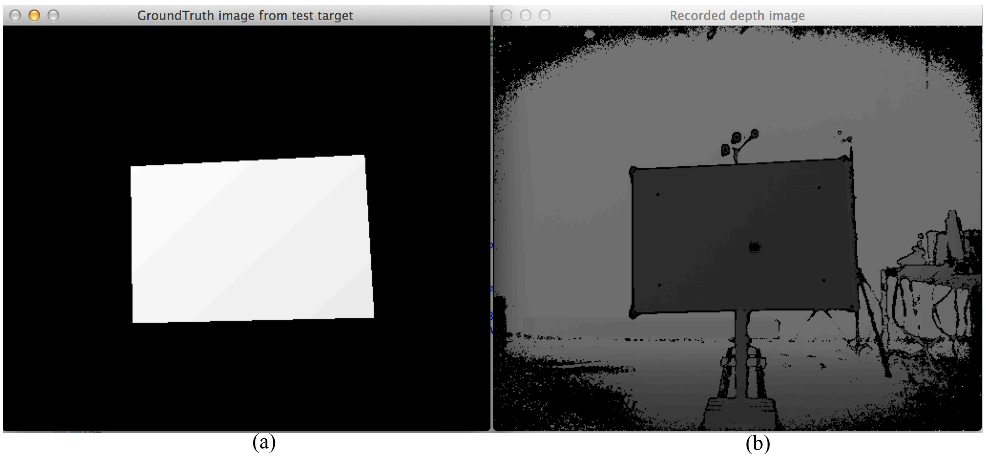
For the 0° position, we used an additional, X=2°, Y=−3.89°, and Z=4° to align the ground truth plane with the recorded image pane. For the 4° position in the right side, we used an additional, X=2°, Y=1.18°, and Z=4° to align the ground truth plane with the recorded image pane. And, for the 5° position in the left side, we used an additional, X=2°, Y=9.89°, and Z=4° to align the ground truth plane with the recorded image pane. Moreover, since the pivot for the depth camera (located at the green marked markers in Figure 4.5(c,d)) is on the surface of the Kinects, we also need to adjust the X-axis and Y-axis position of the Kinect to reach the real physical location of the depth sensor inside the camera. Besides, as the test target is moved far away from the depth camera the offset between the measured depth image and ground truth image tends to increase for Kinect V1 [95]; therefore we needed to additionally adjust the X-axis position accordingly. We subtracted values ΔX and ΔY (in millimeters) from the C_x and C_y values in Equation 4.1 to achieve that. It is worth to mention that while ΔY remained the same through the varying distances of the test target from the depth camera, we needed to increase the value of ΔX (for Kinect V1) as the test target was moved far away from the depth camera.

4.2.3.2 Ground truth data creation

After we obtain the transformed position of the four corners of the test target, we draw a rectangle using `GL_QUADS` and `glVertex3f`. By convention, OpenGL has a right-handed coordinate system this basically means that the positive X-axis is to the right, the positive Y-axis is up and the positive Z-axis is facing the forward direction. However, when we plot a 3D coordinate in OpenGL, the projection matrix switches the direction of the Z-axis; that means we need to reverse the value of Z-axis co-ordinate. After we followed this convention, our transformed corner matrix T_{fin} looks as stated in Equation 4.13. Figure 4.7(a) shows the ground truth image that we plot in OpenGL. Figure 4.7(b) shows the recorded depth image in an OpenCV window. Details about extraction of test target region from other part of the recorded image is discussed in Section 4.2.4.

4.2.4 Extraction of the test target from the ground truth image and recorded depth image

After we create the ground truth depth image of the test target, the next step is to extract the region of interest from the ground truth depth image and the recorded depth image from the depth camera. Since the test target does not fill the whole area of the captured depth image, it contains other objects along with the test target inside it. So, we need to extract only the test target from the depth image and use that image for further calculation for quantifying the amount of noise. The ground truth image that we plot on an OpenGL window also contains other areas outside of the surface area of the checkerboard to fill the window. Therefore, we also need to extract the area, from the OpenGL window showing the ground truth plane, that would contain only the area of the checkerboard's surface area. Therefore, we need a region of interest area detection process that would detect the boundaries of our target object's plane and effectively eliminate other points which does not belong to the plane. For detecting the plane of the test target inside the captured depth image and the plotted ground truth image, we followed the following steps:



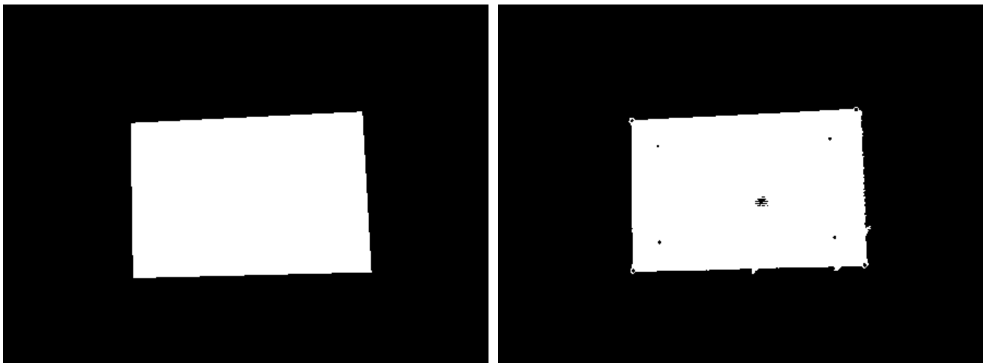
(a) (b)

Figure 4.7 – A side-by-side view of the ground truth image and recoded depth image. Here (a) shows the ground truth image drawn on an OpenGL window and (b) shows the recorded depth image on an OpenCV window. We can see both the images in (a) and (b) are similar in size, shape and orientation.

- Binary image creation
- Bounding box calculation
- Extracting the test target and plotting the extracted target in 3D

Binary image creation

We created binary images for both the depth image captured by the depth camera and the ground truth image which we generated using the calculation stated in the previous



(a) (b)

Figure 4.8 – A side-by-side view of the binary images, using the depth thresholds, of the ground truth image and recoded depth image. Here (a) and (b) show the binary images of the ground truth image and the recorded depth image respectively.

section. The images in Figure 4.8 show the binary images corresponding to the ground truth depth image and the recorded depth image. In Figure 4.8, the white pixels indicate the test target which we used for our experiment i.e. the regions of interest, while the black pixels are the areas which fall outside the region of interest and are ignored eventually. For the captured depth image, we created the binary image by applying a threshold to the depth image. We used an upper and a lower boundary for the threshold that we calculated based on the test target's dimension (width and height) and the distance from the depth camera. For example, if we placed the test target 1.3 meters from the camera, we would put the lower threshold as 1300 millimeters and the upper threshold as 1460 millimeters, see lines 5 and 6 in the Matlab codes in Listing A.1 of Appendix A.1. The upper threshold is bigger than the lower threshold, because the upper part of the test target was a bit tilted in outwards direction from the depth camera. We obtained the upper and lower thresholds by plotting the depth image in 3D in Matlab and then we hovered the mouse over the image to find the lowest and highest depth values for the area covered by the test target. We used the Matlab function `surf`, see line 4 in Listing A.1 of Appendix A.1, which plots a 3D colored surface of an input image. Finally, we apply the threshold to the captured depth image and obtain its binary image, see line 7 in Listing A.1 of Appendix A.1. For getting the binary image of the ground truth image, we used only one threshold that is the value 0 since the areas outside the ground truth plane inside the OpenGL window are black and have pixel values of 0. The code for getting binary image from ground truth image is shown on line 8 in Listing A.1 of Appendix A.1.

Bounding box calculation

After we obtain the binary image of the captured depth image and the ground truth image, we calculate the bounding boxes of the binary images. Later we use these bounding boxes to extract the target object from the captured depth image and the ground truth depth image. As we see from Figure 4.8, the white part of the binary images contain the actual test target, we calculate the bounding box using these binary images. We used the Matlab function `regionprops`, see lines 10 and 11 in Listing A.1, to obtain the exact bounding box of the white part of the binary image (of the captured depth image), which is the region of our interest i.e. our test target. For the process of obtaining the bounding box for the ground truth image, please see lines 15 and 11 in Listing A.1 of Appendix A.1. We use these bounding boxes for the next step that is the extraction of the test targets from the captured depth image and ground truth depth image.

Extracting the test target and plotting the extracted target in 3D

After we obtain the bounding boxes for both the captured depth image and the ground truth depth image, we apply them respectively on the captured depth image and the ground truth depth image to extract the test target from the captured depth image and

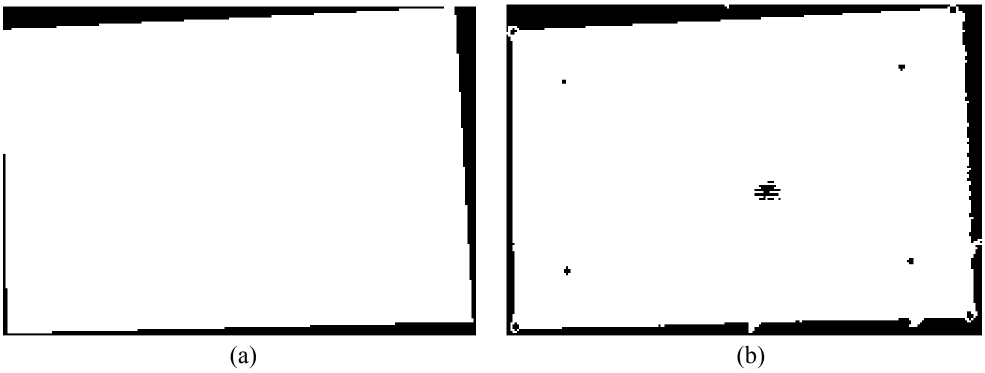


Figure 4.9 – A side-by-side view of the extracted regions from the ground truth image and recoded depth image. Here (a) and (b) show the extracted regions of the ground truth image and the recorded depth image respectively.

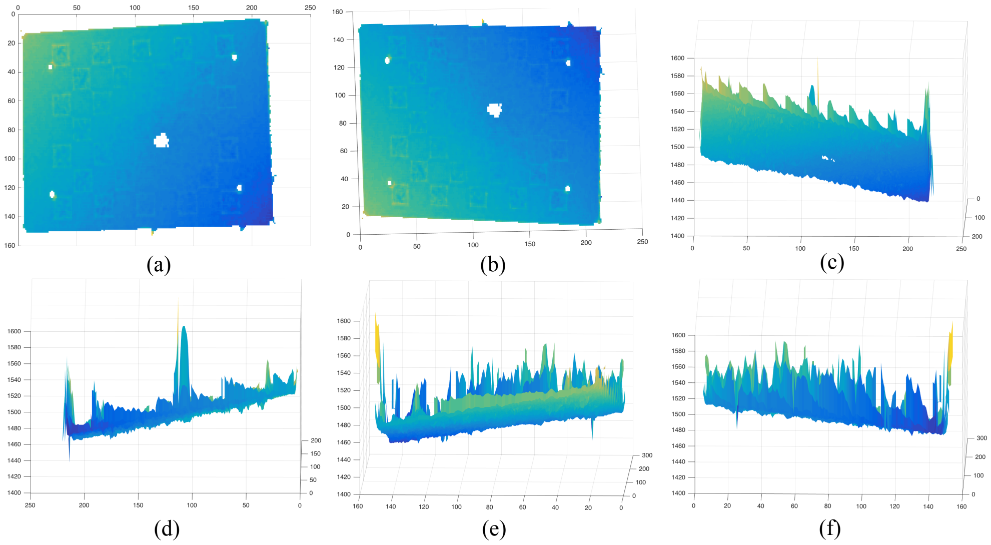
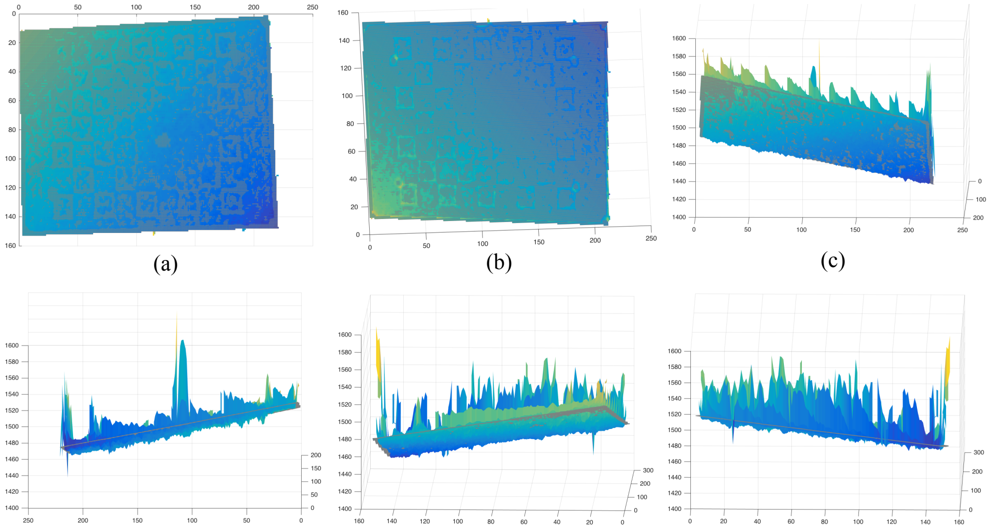


Figure 4.10 – 3D plot showing different views of the test target extracted from the recorded depth image. Here (a)–(f) show the front, back, top, bottom, left and right views of the extracted region of the recorded depth image.

ground truth depth image, see lines 13 and 18 respectively in Listing A.1 of Appendix A.1. Figure 4.9 shows the extracted test target region of the ground truth image and the recorded depth image. We plot these extracted regions in 3D to visualize and analyze the extracted test target area from the ground truth image and the recorded depth image, see lines from 20 to 24 in Listing A.1 of Appendix A.1. Figure 4.10 shows different views of the extracted region of the recorded depth image in 3D. Figure 4.11 shows different views of the extracted region of the recorded depth image and the ground truth image in 3D. We can see that the depth values of the extracted region of the recorded depth image fluctuate



(a) (b) (c)

(d) (e) (f)

Figure 4.11 – 3D plot showing different views of the test target extracted from the recorded depth image and the ground truth depth image. Here (a)–(f) show the front, back, top, bottom, left and right views of the extracted region of the recorded depth image and the ground truth depth image. Here, ground truth image is shown in gray color in (a)–(f). For example, in (c) the ground truth image is the gray surface plane which is passing through the recorded depth image.

from one region to another and the range of the depth values are also not very small. Hence, we fit a plane through these depth values so that we can analyze and compare the difference between the surfaces of the recorded depth image and ground truth depth image. While plotting the extracted target of the recorded depth image in 3D, we noticed that the extracted region is showing unstable movement due to the occurrence of outliers from one frame to another. This unstable nature of the depth image would cause issues to properly calibrate the position of the depth sensor because with an unstable and moving target we would not be able to properly measure at which distance and rotation angle the ground truth would align with the recorded depth image. To resolve this issue and to stabilize the recorded depth image, we used an average image of 90 images instead of using one single image and use that average image to estimate the calibration parameters.

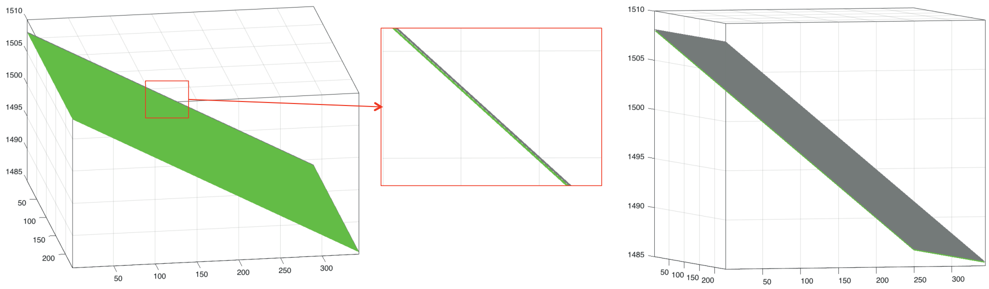
4.2.5 Fitting plane through the ground truth and the recorded depth image

Since the captured depth image contains depth values that vary drastically from one location to another, depending on distance, lighting and viewing angle, we used a plane that would fit the depth values at various locations in the image. This would help us to more precisely measure the difference between the captured depth image and generated ground truth image. We also fit a plane through the ground truth depth image. For the points in a depth image, we calculated a plane with minimal distance to all the points. This was done by using the MATLAB function `fit` that calculates the best coefficients for an equation with given points. We used the polynomial model `poly10` of Matlab to calculate the equation of the plane to be fitted. From the definition of `fittype` parameter of Matlab's `fit` function, for polynomial surfaces, model names have such definition as `polyij`, where i is the degree in x and j is the degree in y. Since, we use model `poly10`, it means that the degree in x is 1 and the degree in y is 0. According to the definition of `fit` function, the plane equation for `poly10` becomes as stated in Equation 4.14; here, $p00$ and $p10$ are the coefficients. We also used options such as `Normalize` and `Bisquare` for normalizing the values and for obtaining a robust plane. The codes for fitting planes are stated in Listing A.2 of Appendix A.2. Figure 4.12 shows the fitted planes of the recorded depth image and the ground truth depth image; here, we can see that both these planes match and the gap between them is very small. The tiny angle (0.2947°) between these fitted planes also shows that the gap between them is very small.

$$z = p00 + p10 * x \tag{4.14}$$

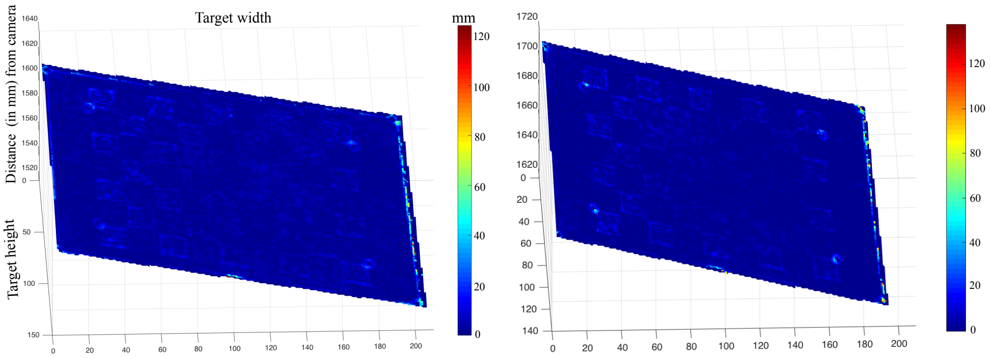
4.2.6 Noise extraction and visualization of the extracted noise

In order to understand the depth sensor noise, generated on the test target, we obtain the noise by subtracting the ground truth image from the recorded depth image. We extract the noise for each measured location of the test target and then we analyze how the noise is changed (or not changed) depending on the viewing distance from the camera, viewing angle of the camera and the lighting condition of the environment. We use



(a) (b)

Figure 4.12 – Front and back view of the fitted planes through the extracted regions of the recorded depth image and the ground truth depth image; green and gray planes represent the recorded depth image and the ground truth depth respectively. Here (a) and (b) show the front and back views of the fitted planes of the extracted region of the recorded depth image and the ground truth image respectively. The zoomed image in the middle shows the very small distance between these planes.



(a) (b)

Figure 4.13 – Extracted noise mapped on the ground truth image at two different distances. Here (a) and (b) shows the noise mapped on the ground truth images created at distances 1.6 meters and 1.7 meters respectively. The color bar shows the scale of the amount of noise in millimetres (mm) over the surface of the ground truth images. We can see that the variation of the amount of noise in image (a) and (b) which are located at two different positions.

the Matlab function `imabsdiff` to get the difference between the recorded depth image and the ground truth image. Then we map the noise to the ground truth image so that we can perceive the quantity of noise on the surface of the ground truth image. This mapping is done to perceive the effect of noise on the ground truth image and analyze it for understanding the noise characteristics. The Matlab codes for extracting the noise and mapping it onto the surface of the ground truth image is stated in Listing A.3 of Appendix A.3. Figure 4.13 shows an example of the extracted noise mapped onto the ground truth images that are created at two different distances from the position of the depth camera. More results for different distances, viewing angles and lighting conditions are shown in the next section.

4.3 Results and discussions

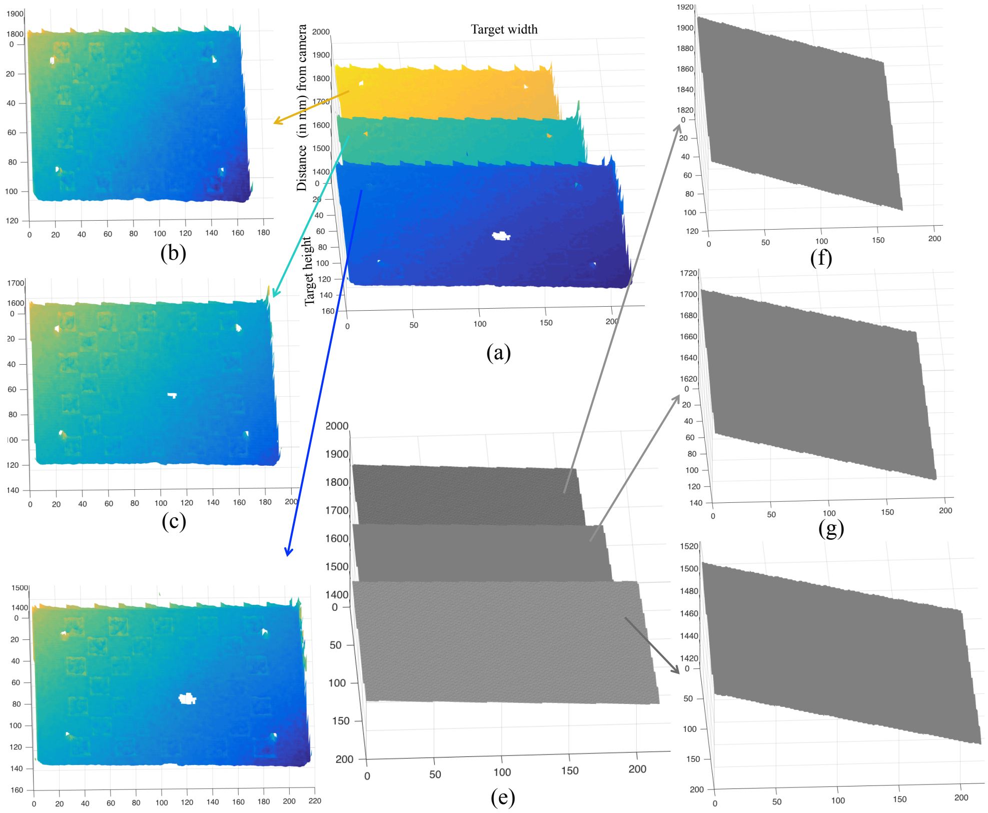
Here we discuss the various results that we obtained throughout different steps of the sensor noise extraction and visualization experiment. We first show how the depth image changes depending on the distance of the test target from the depth camera. In addition to this, we show the corresponding ground truth images for the different distances and the noise extracted and mapped on the ground truth image. Then we present visualizations of the depth image and the ground truth image from three different angles for the same position to understand how the viewing angle affects the depth images and the corresponding noise on them. Finally, we present and compare the depth images captured in two different lighting conditions and try to understand how different lighting sources affect (or, do not affect) the noise characteristics of the depth images. We also compare the depth images captured by Kinect V1 and Kinect V2 and analyze the noise characteristics from these two different depth sensors (Kinect V1 and V2 use structured light sensing and ToF methods respectively to measure the depth of an object).

4.3.1 Visualization of depth images and ground truth images at varying distances

Here we show the appearance of noise on the surface of the recorded depth images at various locations of the test target from the depth camera. Figure 4.14 shows 3D views of the recorded depth images of the test target at three different distances (1.4 meters, 1.6 meters, and 1.8 meters) from the depth camera. We know from the working principle of Kinect V2 that the accuracy of the depth estimation varies with the variation of an object's distance from the depth camera [95, 96]. Similarly, we see that as the target is moved to different distances from the camera, the quantity of noise on the surface of the depth image also varies. It becomes more clear by looking at the four holes, on Figure 4.14(b) to (d), which occur at the positions where we placed four reflective objects so that we get some noise at these locations. We also perceive a rather large hole in the middle of the images in Figure 4.14(c) and (d) which is caused by an unwanted reflection on the test target's surface. As we move further from the camera, the reflection gets smaller and so the size of the hole; in (b), we do not see this hole since at this position that part of the target's surface is not reflective anymore. We have also depicted the ground truth image along with the recorded depth image at the mentioned distances in Figure 4.15. It shows how the ground truth is aligning and intersecting with the recorded depth images over the surface area. In order to visualize the noise, we have subtracted the ground truth image from the recorded depth image and showed the noise at those distances in Figure 4.16; here too, we can see that the amount of noise changes as the target is moved far away from the depth camera. Figure 4.20(a) shows how the noise increases or decreases with the increase or decrease of the distance of the target from the camera.

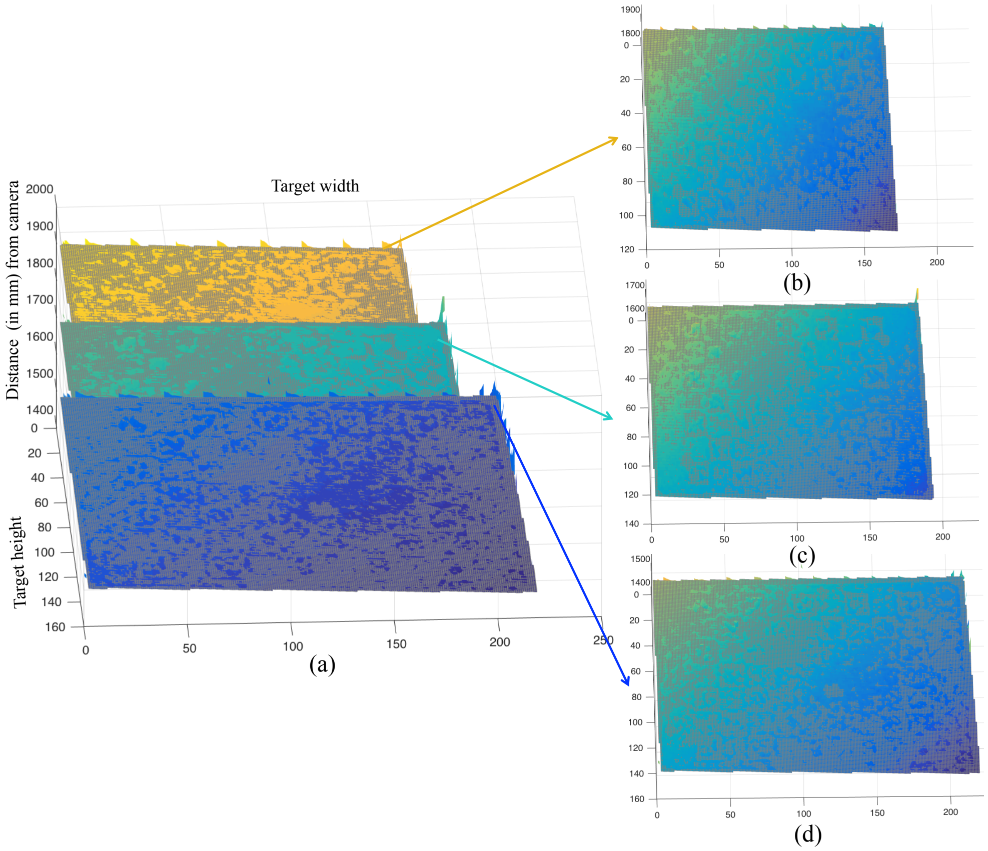
4.3.2 Visualization of depth images and ground truth images at varying viewing angles

We present here the effect of changing the viewing angle of the depth camera with respect to the test target on the quantity of noise. We used three viewing angles of the depth camera – first with 5° rotation of the camera to the left side, then at 0° rotation of the camera i.e. the camera is kept at perpendicular direction with respect to the test target and finally, at 4° rotation to the right side. Figure 4.17 shows the recorded depth image and ground truth image of the test target for those three viewing angles. Here, we can see variations in depth values from one viewing angle to another. At the same time, we also observed a variation in the angle between the fitted planes that pass through the recorded



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

Figure 4.14 – 3D views of the recorded depth images and ground truth images of the test target at three different distances from the depth camera. Here, (a,e) show the 3D views of all the depth images and the ground truth images in one plot whereas, (b,c,d,f,g,h) show different plots showing different 3D views of the test target located at distances 1.4 meters, 1.6 meters and 1.8 meters from the depth camera respectively. We can see presence of a rather large hole in the middle of the depth images in (c) and (d) which were caused by reflections. As we move far away from the lighting source causing the reflection, the hole gets smaller and in (b) we do not perceive that hole anymore. We perceive a difference in the quantity of noise on the surface of the depth images as the target is moved farther from the camera by looking at the four holes (we placed four reflective objects on these positions so that we get some noise at these locations). By looking at images (d) to (b), we can see that the amount of noise varies (noise increases as the distance of the object from the camera increases) with varying distances of the test target from the camera.



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

Figure 4.15 – 3D views of the alignment of ground truth image and recorded depth images of the test target at three different distances (1.4 meters, 1.6 meters and 1.8 meters) from the depth camera. As we move from (d) to (b), we can perceive how the ground truth aligns and intersects with the image plane of the recorded depth images.

depth image and ground truth image respectively, see Figure 4.18. Table 4.1 shows the change in the angle between the fitted planes through the recorded depth image and the ground truth image for three viewing angles. We also perceive the variations in the amount of noise from one rotation angle to another, see Figure 4.19. Here, we mapped the extracted noise on the ground truth image as color-coded texture and hence we can perceive the scale of noise on the surface of the ground truth image; the noise is obtained by extracting the ground truth image from the recorded depth image. We perceive more variation of noise on the right edge of the test target when we change the viewing angle of the camera. The scales shown on the right side of each plot in Figure 4.19 also indicates that the noise quantity changes from one angle to another. Besides, the color-coding of the noise at the right edge also changes from one image to another which indicates the

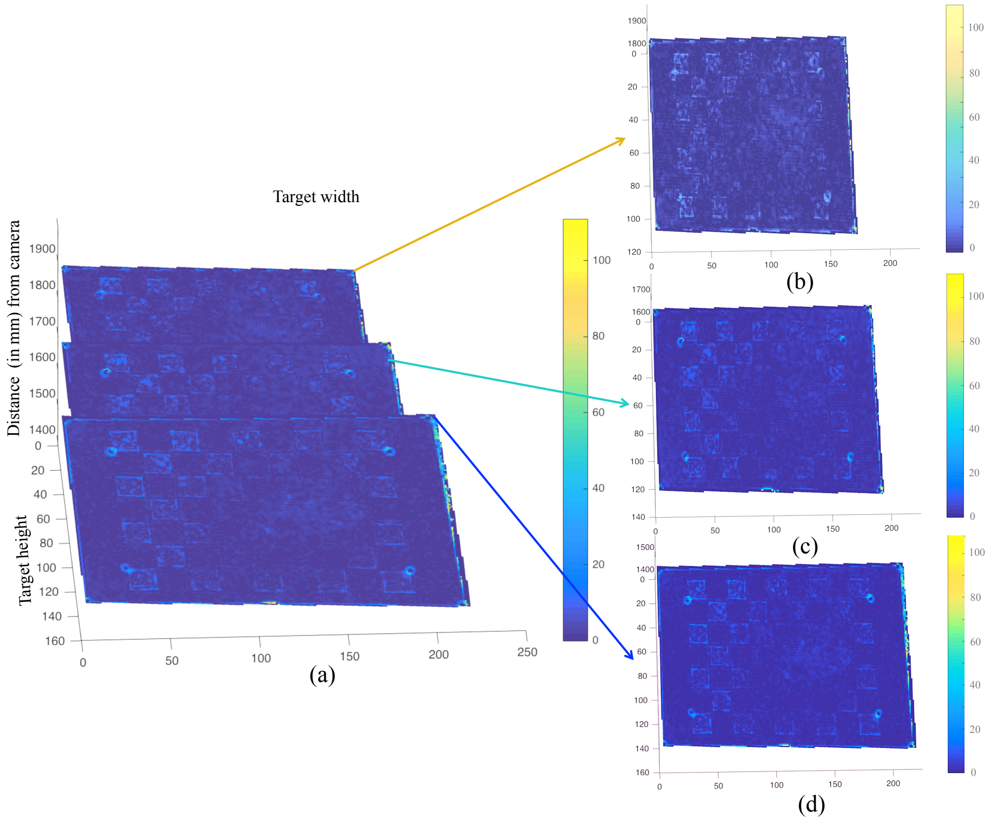
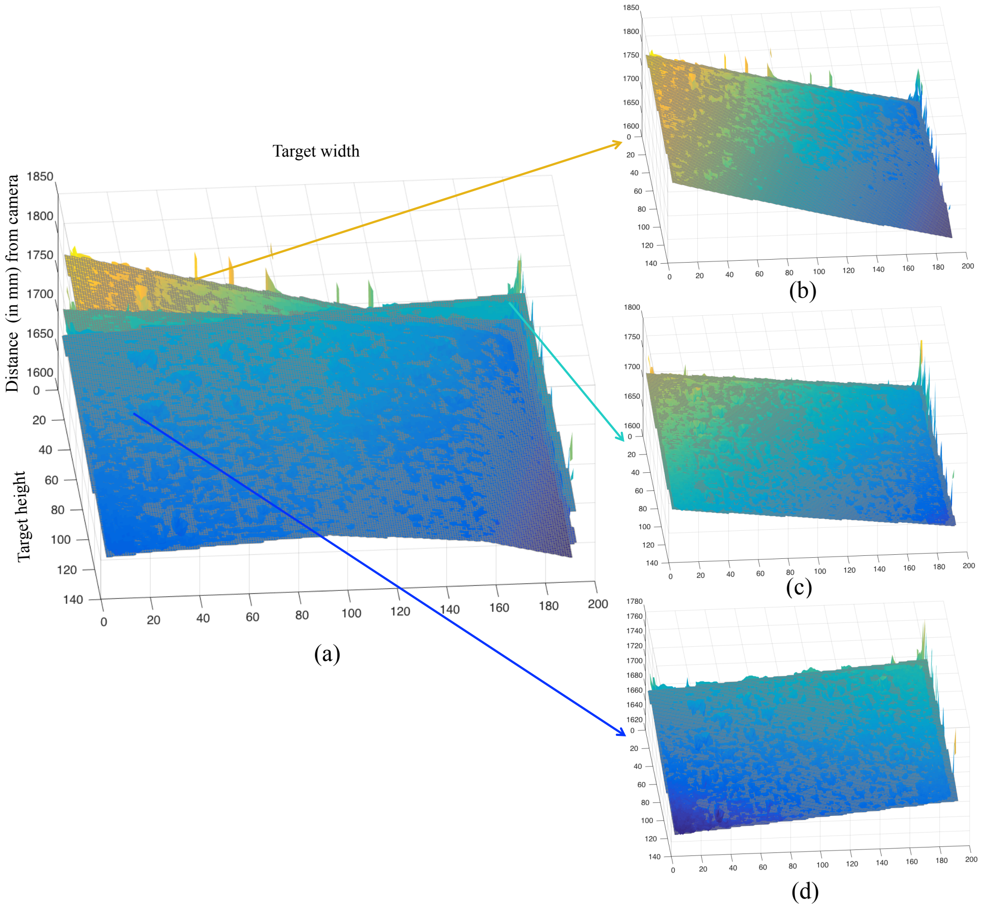


Figure 4.16 – 3D views of the noise obtained by subtracting the ground truth images from the recorded depth images of the test target at three different distances (1.4 meters, 1.6 meters and 1.8 meters) from the depth camera. Here, by looking from (d) to (b), we can see that the amount of noise changes as the target is moved far away from the depth camera.

change of noise amount. Figure 4.20(b) shows how the noise increases or decreases with the increase or decrease of the viewing angle of the camera with respect to the target.

Table 4.1 – Change in the angle between the fitted planes through the recorded depth image and the ground truth image for three viewing angles.

Rotation angle of the depth camera	Angle between the fitted planes
5° to the left side	0.5185
0°	0.5907
4° to the right side	0.5790



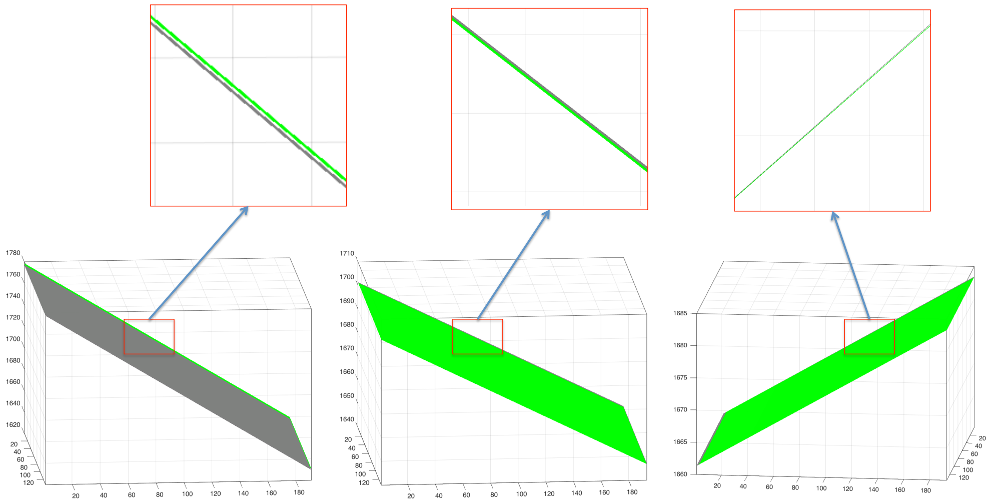
(a)

(b)

(c)

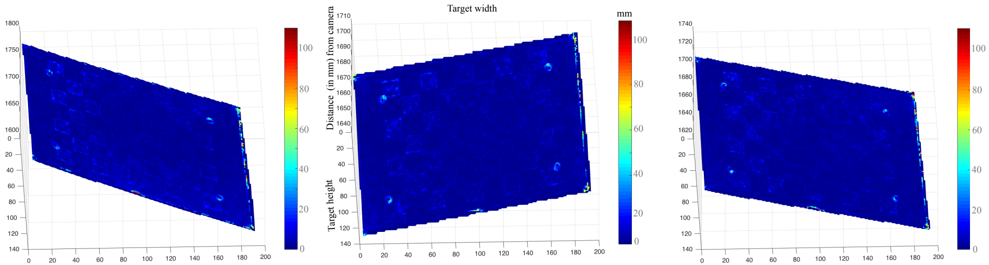
(d)

Figure 4.17 – 3D views of the recorded depth image and ground truth image at three different viewing angles at one specific distance from the depth camera. Here, (a) shows the plots from all the three viewing angles and (b,c,d) show the plots for the rotation of the depth camera to 5° to the left side, 0° (no rotation i.e. perpendicular to the test target) and 4° to the right side respectively. We see here variations in depth values on the surface of the test target from one viewing angle to another.



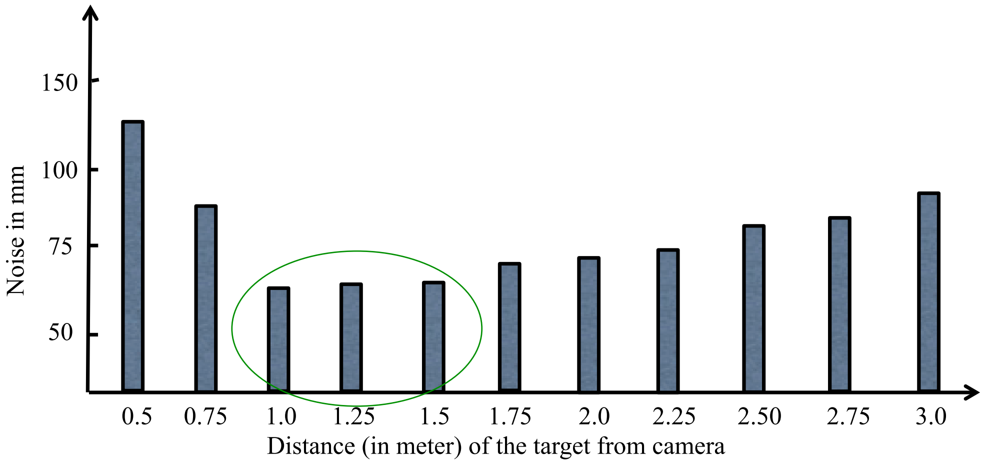
(a) (b) (c)

Figure 4.18 – 3D views of the front and back side of the fitted planes through the extracted regions of the recorded depth image and the ground truth depth image at the three angles, mentioned in Figure 4.17, respectively; green and gray planes represent the recorded depth image and the ground truth depth image respectively. The zoomed images above (b,c,d) show how the fitted planes align with each other. Here, the zoomed parts of (a) and (b) show a small gap (less than 0.5 millimetres) between the fitted planes of the recorded depth image and the ground truth image, whereas (c) show no such distance. This gap occurs due to manually moving the test target from one location to another or manually rotating from one angle to another which seems to induce such gap.

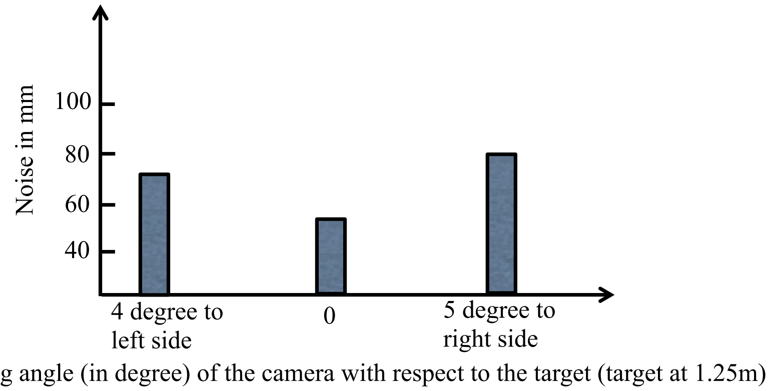


(a) (b) (c)

Figure 4.19 – 3D views of the noise obtained by subtracting the ground truth images from the recorded depth images of the test target at three different viewing angles, mentioned in Figure 4.17, respectively. Here, by looking from (a) to (c), we can see that the amount of noise varies from one viewing angle to another, specifically at the bottom right corner of the test target shows the variations in the amount of noise in millimeter (mm). Moreover, the scales (shown at the right side of each images) in these three images have different maximum values which also indicate that the amount of noise varies from one angle to another.



(a)



(b)

Figure 4.20 – Variations of the noise quantity at various distances of the object from the camera and at various viewing angle of the camera with respect to the target. Here, (a) and (b) show the amount of noise changed due to the change in distance of the target the camera and for the change in viewing angle of the camera. In (a), we can see that the lowest noise (marked with a green ellipse) occurs in between the range of 1 meter and 1.5 meters and it increases beyond this range. The error is lowest at this range because at this distance range, Kinects can estimate the depth values better. In (b), we can see that the lowest error occurs when the viewing angle between the camera and the target is 0 and then it becomes bigger with greater viewing angles.

4.3.3 Visualization of depth images and ground truth images at varying light sources

For testing the effect of various light sources on the amount of artifact generated on the surface of the target, we used two light sources mounted on the rig to which the tracking cameras are attached. In one source we had room lights which have a color temperature of $5,500$K and in another, we used four studio lights which have a color temperature of $6,500$K each. We observed slight variations of the test target's depth values under these lighting sources. Although the variations are not huge, there are slight variations in the depth values. Figure 4.21 shows the color images, depth images and the corresponding 3D plot of the extracted region of the target and the ground truth image for the two lighting sources. Although the color image shows different shades of lighting, we perceive a slight change in the depth values both in the depth images and on the 3D plots. To visualize the difference between the depth image, we show a plot in Figure 4.22(a) which depicts the absolute differences between these images which also indicate tiny variation among these two images. The noise difference in Figure 4.22(b) also indicate similar observation. It is worth to mention that we obtained the noise for these images by subtracting the ground truth images from the recorded depth images. For a better quantification of the noise per pixel in these images, we depict, in Figure 4.23, bar diagrams which show the mean noise per pixel for these images. Here too, we perceive the variations in the noise level for these two images which are not very drastic for the majority of the pixels.

4.3.4 Observations from the experiments with Kinect V1 and V2 sensors

There are a few other issues which we observed while conducting these experiments. Firstly, we noticed, for Kinect V2, a variation in depth values for the different colored squares with black and white color. Since all these squares are on the same plane they should all have the same depth value and should not appear as different squares on the depth image. This artifact has already been discussed in [96]. Figure 4.24 (and Figure 4.11(a) and (b)) show examples of such artifacts. Kinect V1 sensor do not exhibit such artifact.

For conducting the experiment with Kinect V2 sensor, we needed to run Kinect V2 for 25 minutes before recording any scene. This waiting time helped us to get a steady depth value of the test target as before that time the measured depth value fluctuates a lot. The distance measurement method of Kinect V2 has a strong correlation with the temperature of the device, whereas Kinect V1 has a weak correlation [95]. Therefore, we waited approximately 25 minutes to avoid temperature-related influences on the depth images. For Kinect V1, a short warm-up time was fine.

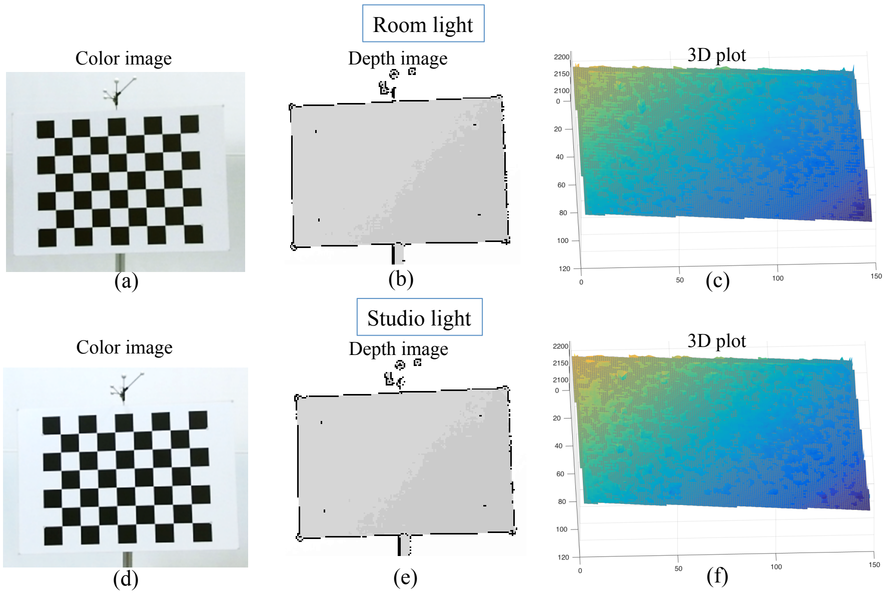


Figure 4.21 – Recorded depth image and ground truth image under two different lighting sources. Here, (a) to (c) show the color image, depth image and the 3D plot of the extracted region of the target and the ground truth image under room light (with color temperature of 5, 500K). And, (d) to (f) show the color image, depth image and the 3D plot of the extracted region of the target and the ground truth image under studio light (with color temperature of 6, 500K). Under these two lighting sources, although the color images show different shades of lighting, very little change is perceived in the depth images and on the 3D plots.

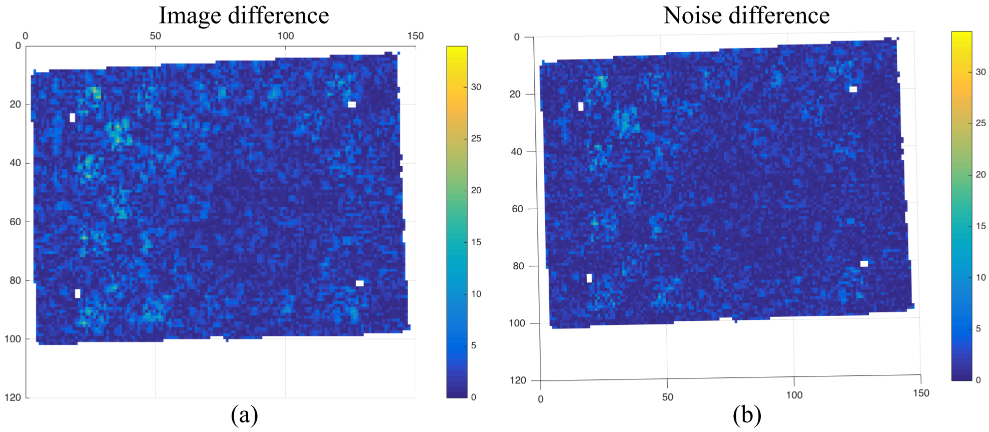


Figure 4.22 – Absolute difference between the depth images under two lighting sources and between the noise on these two depth images. Here, (a) shows the absolute differences between the depth images under room light and studio light. By looking at the plot and the scale of variation on the right side, we perceive very small variation among these two images. And, (b)shows the absolute differences between the noise generated on the depth images under room light and studio light. Here too, we observe very little variation among the two noise quantities.

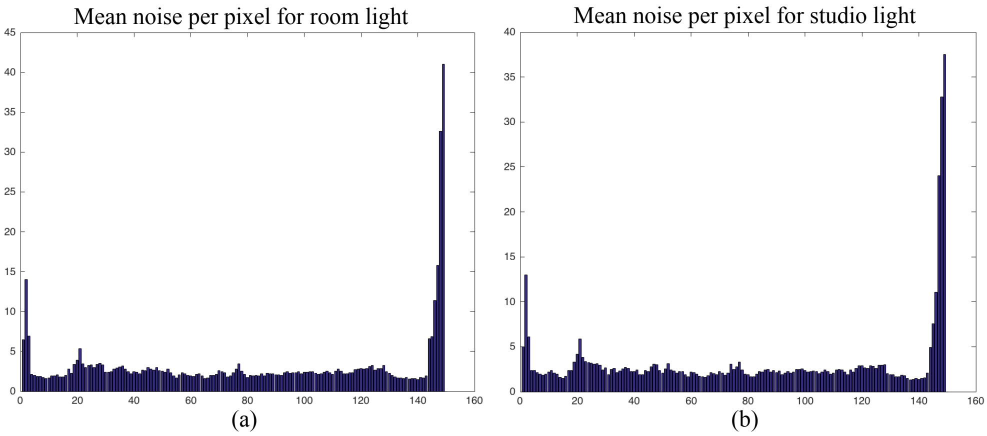


Figure 4.23 – Bar diagrams showing the mean noise per pixel for the images recorded under room light and studio light. Here, (a) and (b) show the bar diagrams of the mean noise per pixel of the depth images under room light and studio light respectively. Although, we perceive little variations in the noise level for these two images, but the variations are not very drastic for the majority of the pixels.

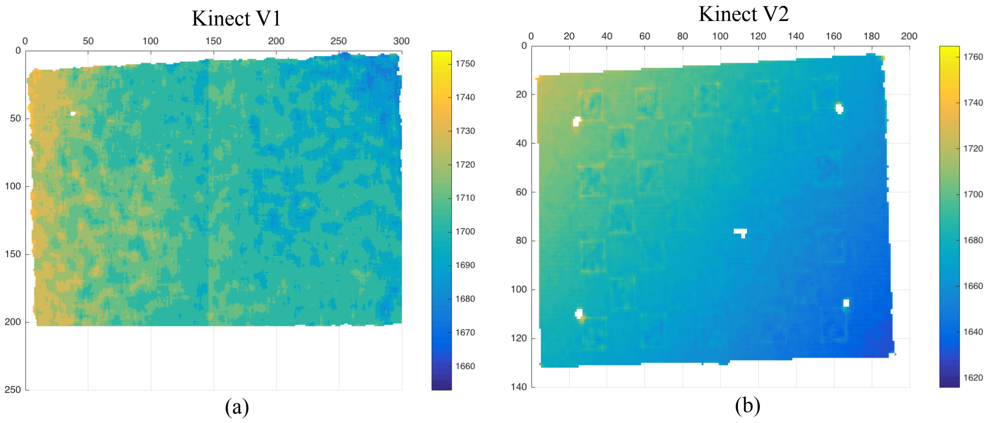
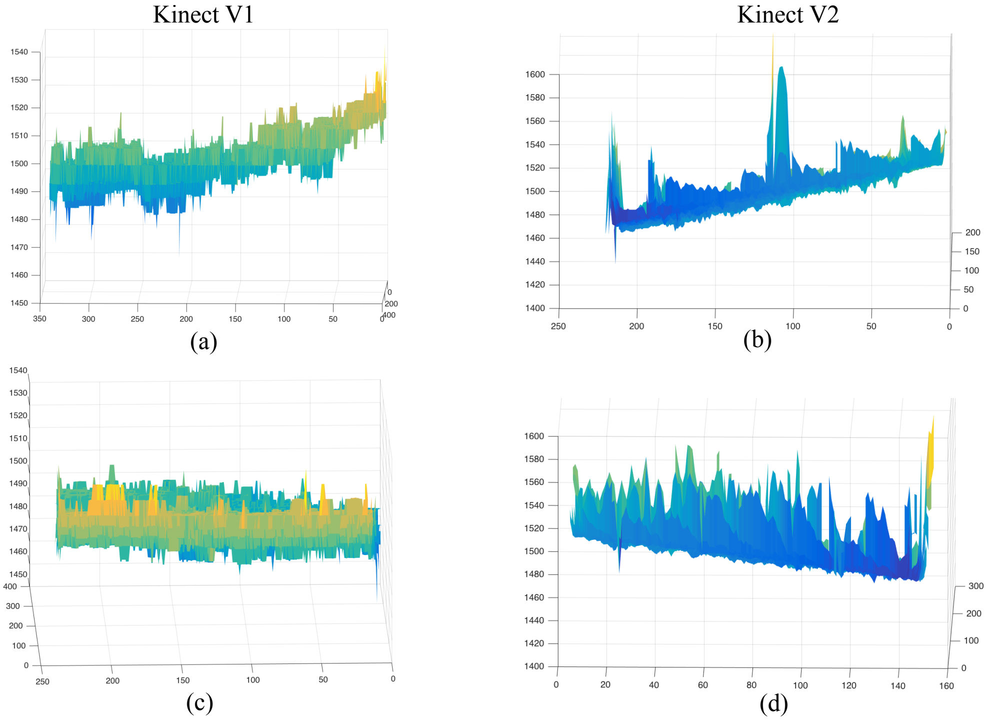


Figure 4.24 – Extracted test targets, located at the same position, from the depth images captured by Kinect V1 and Kinect V2 sensors. Here, (a) and (b) show the test targets by Kinect V1 and V2 respectively. We can see the squares in (b) (also in Figure 4.11(a) and (b)) which correspond to the actual black and white squares on the test target (checkerboard) we used for the experiments. Although these squares are on the same plane, Kinect V2 is showing different depth values for these squares. We don't see such squares on the depth image of Kinect V1 in (a).



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

Figure 4.25 – Top and bottom views of the extracted test target in 3D showing the depth values. Here, (a) and (b) show the 3D views of the top side of the test target extracted from the depth images captured by Kinect V1 and V2 respectively. We can see that the depth values are fluctuating within a relatively short range for Kinect V1 than for Kinect V2. For Kinect V2, we see quite large fluctuation of depth value compared to the fluctuation in Kinect V1. We also perceive similar behaviour of the depth values for the bottom view of the extracted test target in (c) and (d). Here, we can see even more spikes indicating more fluctuations in the depth values for Kinect V2 than Kinect V1.

While conducting the experiments with Kinect V1 and V2 sensors, we came across different levels of outliers on the depth images captured by these two sensors. For Kinect V1, the outliers are more close to each other whereas the outliers for Kinect V2 are spread out from one another forming spike shapes. Figure 4.25 shows an example of such an artifact where (a) and (b) show that the depth values are fluctuating within a relatively short range for Kinect V1 than for Kinect V2. For Kinect V2, we see quite a large fluctuation of depth value compared to the fluctuation in Kinect V1. We also perceive similar behavior of the depth values for the bottom view of the extracted test target in Figure 4.25(c) and (d). Here, we can see even more spikes indicating more fluctuations in the depth values for Kinect V2 than Kinect V1. Since the depth images from Kinect V2 contain a rather large quantity of outliers, our proposed depth enhancement strategy, discussed in Chapter 3, would essentially be able to remove those outliers and stabilize the depth values for each pixel.

4.4 Future works

Due to the time constraints, we were not able to test the noise characteristics of other depth cameras. In the future, we would like to carry out the experiment with other depth cameras and analyze the noise characteristics for those depth cameras. Moreover, we would like to perform interpolation using the calibration results from the experiment so that we can effectively estimate the depth values for dynamic scenes with more precision. Instead of using a rectangular target, we would use a sphere to get all possible orientation of the test target at one capture and use more light sources to cover all possible variations of distortions that occur on a depth sensor. We would move the sphere in different vertical and horizontal places, rather than moving along one line as we did in this experiment, to get a better interpolation and hence a better approximation of the depth value for any given position and orientation of a target object in a scene. Besides this, we would also like to dynamically select the number of frames in the sliding window of the depth enhancement strategy discussed in Chapter 3 which would help us to better optimize the whole processing pipeline of the enhancement strategy.

Chapter 5

Reduction of transmission data

This is the second focus of this thesis where we discuss our proposed data reduction approaches. Since the captured data needs to be transmitted to some location/device from the acquisition device, in case of a large amount of captured data, there is always a demand for reduction of the input data that does not contribute to the final output. An efficient reduction strategy not only reduces the data transmission latency, but also makes the data processing less complex and less time-consuming. Besides this, there is industry requirement which states that a method's generated data should use less memory and consume the minimum amount of data to ensure smooth transportability of the data from one location to another [1]. To fulfill these requirements, we propose two data reduction methods for reducing the amount of image data without compromising the quality of the final output. The works in this chapter is published in [97, 98].

5.1 Problem description and motivation

While a moderate amount of processing time and power are required to process the data generated by a single commodity depth sensor such as Microsoft Kinect or ASUS Xtion Pro, we can easily imagine the required processing power to handle the massive volume of data which is produced when multiple cameras are used in parallel to capture a scene from various angles. In recent years, we have seen that quite a few applications, such as telepresence [32, 99], tend to be equipped with multiple cameras (see Figure 5.1) to transmit and display the whole scene of the communication space. When a multi-camera setup is used for capturing a scene to support multi-view 3D reconstruction of the scene, the large volume of generated data normally requires additional processing for the final output. These large data need to be processed at the acquisition site, might need aggregation from different network nodes and transmitted to the receiver site. Transmission of this huge data from multiple cameras becomes critically challenging when they are sent over a limited bandwidth network. Even when a single camera is in use in

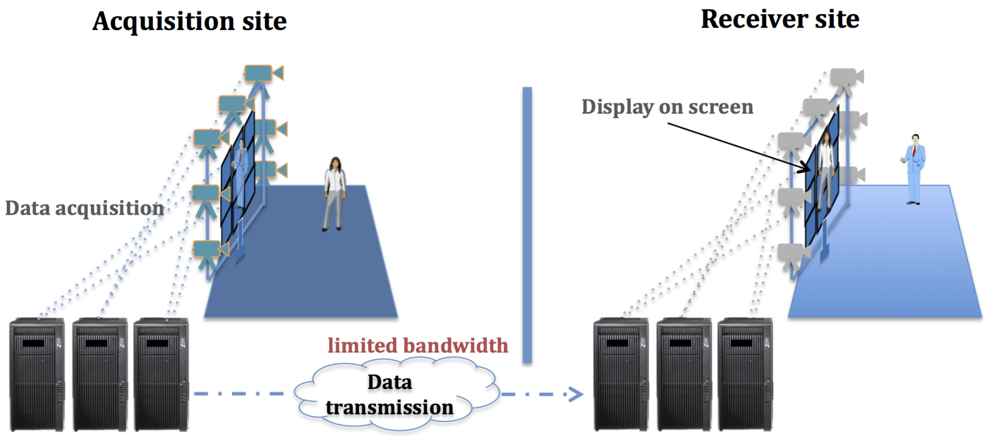


Figure 5.1 – A communication system with multi-camera setup.

an area where network bandwidth is low and unpredictable, such as in a disaster location where a robot is capturing the scene to transmit it to the service station for processing the data to take action accordingly, the captured data becomes too large for the network to transmit properly and hence, need efficient reduction. Compression of these large data, without further consideration, might resolve this issue to a certain degree, but such compression degrades the quality of the data.

We know that depending on the final viewing-angle of a 3D reconstruction or the users' focus on a reconstructed scene, not all data from every camera contribute equally to the final output. Hence, usually, there are camera data from certain cameras that can be discarded for a particular viewing angle of a scene. Besides, based on a user's focus on a scene, data that falls outside of a user's gaze, can also be discarded. Hence, the improvement of coding efficiency at low bitrates before applying the classical compression methods, rather than only applying those compression methods, would be useful in transmitting such data.

Moreover, although our proposed depth data enhancement framework, described in Chapter 3, does not rely on the accompanying color images, many computer vision applications use the color and depth image pairs to reconstruct a colored 3D model. Thus, the combination of color and depth image streams results in quite a heavy amount of data that needs to be processed for generating the final output. Often, not all the color data on all the color frames are required for the final output; hence, we can also find removable color data in the color image stream.

Therefore, to address the above mentioned issues with massive camera data, we propose two data reduction strategies: one for multi-camera data capture setup where certain part of a camera's data can be removed and another for color image data (acquired by multi-camera setup) reduction where certain color data in certain color frames can be discarded. Both of these strategies are aimed towards reducing the overall data required for certain applications and subsequently, reducing the amount of processing power needed to yield the final output. It is worth to mention that both of these strategies are considered as preprocessing steps before applying classical compression techniques; they are not meant to apply as alternatives of any data compression methods that in fact cover a completely different area of work than ours.

5.2 Related works in camera data reduction

We discuss the related work in camera data reduction here, instead of discussing it in Section 2.2, because this is a secondary focus of the thesis which wanted to keep separate from the main focus of the thesis.

The approaches pursued in camera-data reduction use a variety of approaches to reduce the amount of data captured with multi-camera setups. These approaches can be divided into two broad categories. In the first category, the approaches mainly focus on reducing the data volume in such a manner that they require lower bandwidth to be transmitted to other locations/devices. Most of these works apply a dynamic camera selection strategy or dynamic data stream selection strategy. In the second category, the approaches mainly focus on transmitting down-sampled non-key-frames (NKF)s of each camera so that they require less bandwidth and later at the receiver end, they enhance those down-sampled NKFs by using Super Resolution (SR) methods.

For instance, Willert et al. [100] propose a dynamic camera selection strategy which reduces the number of recording cameras at the acquisition site. Here, they propose to use data from only those cameras which capture the scene from the perspective of the user. They also propose to use a dynamic frustum selection method when the dynamic camera selection fails. In another work [101] Lamboray et al. classify the camera data stream into several categories such as bulk data, sporadic-event data, and real-time streaming data. Based on the positional information, obtained through a *backchannel* between acquisition and receiver location, of the viewing user (located at a different location) they apply different strategies to transmit selective updates of the scene from the acquisition location. The effectiveness of such selective transmission of camera data has been studied in the work by Maimone et al. [102] where the authors suggest that when the change does not take place in all parts of a scene, camera selection should focus on the reduction of the overall amount of data. Lien et al. [103] propose another notable data reduction strategy where they apply an approach based on model-driven data reduction strategy.

On the other hand, Brandi et al. [104] propose to use SR methods, at the receiver site, to enhance the down-graded NKFs. Here, they down-sample the NKFs and keep the Key Frame (KF)s at original resolution; encoding is performed on this combination of degraded NKFs and original KFs. By employing this, they gain a better video compression rate by means of up-sampling those down-sampled NKFs at the receiver site. The works by Shen et al. [105] and Hu et al. [106] also use a similar approach but by using an exampled-based SR method and an adaptive SR method respectively to upgrade the NKFs. There are a few other spatial color image enhancement methods [107, 108], other than the SR methods, which are computationally very expensive and are not suitable for our problem domain.

Recently, we have seen usage of very Large High Resolution Display (LHRD)s which are mainly used to display 3D reconstruction of a scene captured with multiple cameras. One such LHRD is a light field display. Among the notable works which propose data reduction strategy for such LHRD with a multi-camera setup, Jones et al. [109] propose a set of rendering methods for an autostereoscopic light field display which is able to present interactive 3D graphics to multiple simultaneous viewers covering 360 degrees around the display. They apply a multiple-center-of-projection rendering technique for creating perspective-correct images from arbitrary viewpoints around the display. In [110], Magnor et al. present another approach where they apply vector quantization, DCT coding and transform coding using spherical functions to the light field compression technique. Here, the first coder decodes the recorded light-field segments very fast and thus achieves interactive rendering rate and then the second coder works as disparity compensating coder which incrementally refines the light field during the decoding and predicts the intermediate light field images.

5.3 Data reduction by efficient degradation and enhancement of color frames

Using classical compression methods to reduce the large volume of data generated by multi-camera setups might solve the problem to a certain degree, but using such compression techniques directly on the acquired data does not guarantee efficient data reduction. Reducing the input data from dynamically selected frames which can be enhanced at a later stage, would be beneficial for transmitting the camera data in real-time. To that end, we propose a camera data reduction strategy which supports efficient bandwidth usage for the frames of a multi-camera setup. Our approach improves the coding performance by means of degrading the color information of NKFs at the acquisition site; we do not degrade the KFs. Unlike the methods described in [104–106] which reduce the resolution of the NKFs, we reduce the amount of color data of the NKFs which aids to reduce the image data. We also propose two different methods which we use to enhance, at the receiver site, the degraded NKFs by using the information stored in non-degraded KFs. Thus, our degradation method acts as a pre-process to the classical video compression methods while our two post-processing methods enhance the image quality of the encoded frames at low bitrates. Our strategy reduces the encoder complexity, improves the compression rate and aids to transmit large amounts of data from multiple cameras at low bitrate.

5.3.1 Proposed method

Our problem domain consists of an acquisition site, a receiver site, a display screen on both sites and cameras placed at certain positions to capture the scene; see Figure 5.1. The captured frames are sent, from the acquisition site, through the available network to be displayed on the screen at the receiver site. Here, the communication works in both ways; we used the terms *acquisition* and *receiver* site for the ease of explanation. As we mentioned previously, our proposed approach consists of a degradation process of the NKFs which acts as a pre-processing step and two enhancement processes which enhances the degraded NKFs with the information from non-degraded KFs. In the next subsections, we discuss the details about our degradation and enhancement strategies in detail.

5.3.1.1 Degradation process

Unlike the methods in [104–106], we degrade the frames not by reducing the resolution but by reducing the color information. We transmit the KFs with full color information and the NKFs with reduced color information which eventually improves the compression rate of the encoder. The diagram in Figure 5.2 shows our basic degradation model.

At first, we convert the color space, from RGB to CIELAB, for each of the NKFs, because CIELAB provides independent access of the lightness and color information, and it is more uniform than RGB [111]. We get the corresponding L*a*b* values of each NKFs after this conversion. Then we apply Principle Component Analysis (PCA) to de-correlate the image data in the CIELAB space. For color degradation, we do not change the information of the L* channel and work only with a* and b* channel to access the chromaticity parameters. We apply the following formulae, in Equation 5.1 and 5.2, to obtain the eigenvectors and eigenvalues.

$$
\begin{gathered}
P_{in} = I - I_m, \quad C_v = P_{in}^T P_{in} \\
E_{vc} = \begin{bmatrix} E_{vc11} & E_{vc12} \\ E_{vc21} & E_{vc22} \end{bmatrix}, \quad E_{vl} = diag(sort(\begin{bmatrix} E_{vl1} \\ E_{vl2} \end{bmatrix}))
\end{gathered}
\tag{5.1}
$$

In Equation 5.1, P_{in} = independent/de-correlated color channel axis, I = NKFs, I_m=mean of I, C_v = covariance matrix, P_{in}^T= transpose of P_{in}, E_{vc}= eigenvector matrix and E_{vl}= eigenvalue matrix. Here, P_{in} and I are raster image matrix in CIELAB color space. Since we apple PCA on two variables, i.e. on two channels (a* and b* channels) of CIELAB space, we get four eigenvectors E_{vc11}, E_{vc12}, E_{vc21} and E_{vc22}. For the same reason, we get two eigenvalues E_{vl1} and E_{vl2}.

Then, we project the E_{vc} on P_{in} and get the reduced axes (the two Principle Component (PC)s P_{in1} and P_{in2}), which contain reduced color information of the a* and b* channels, according to the formula in Equation 5.2.

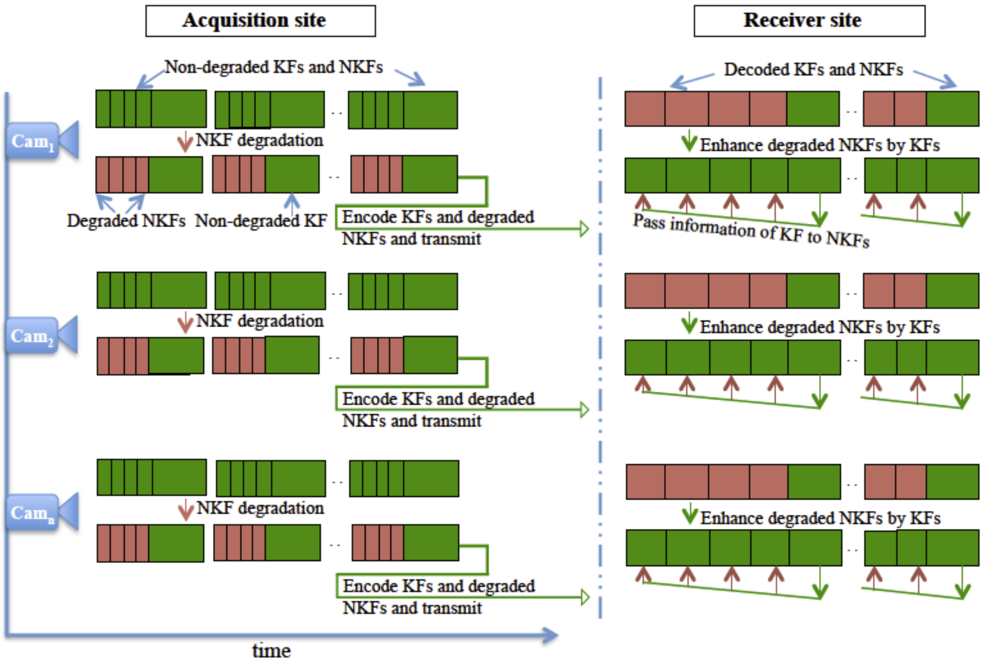
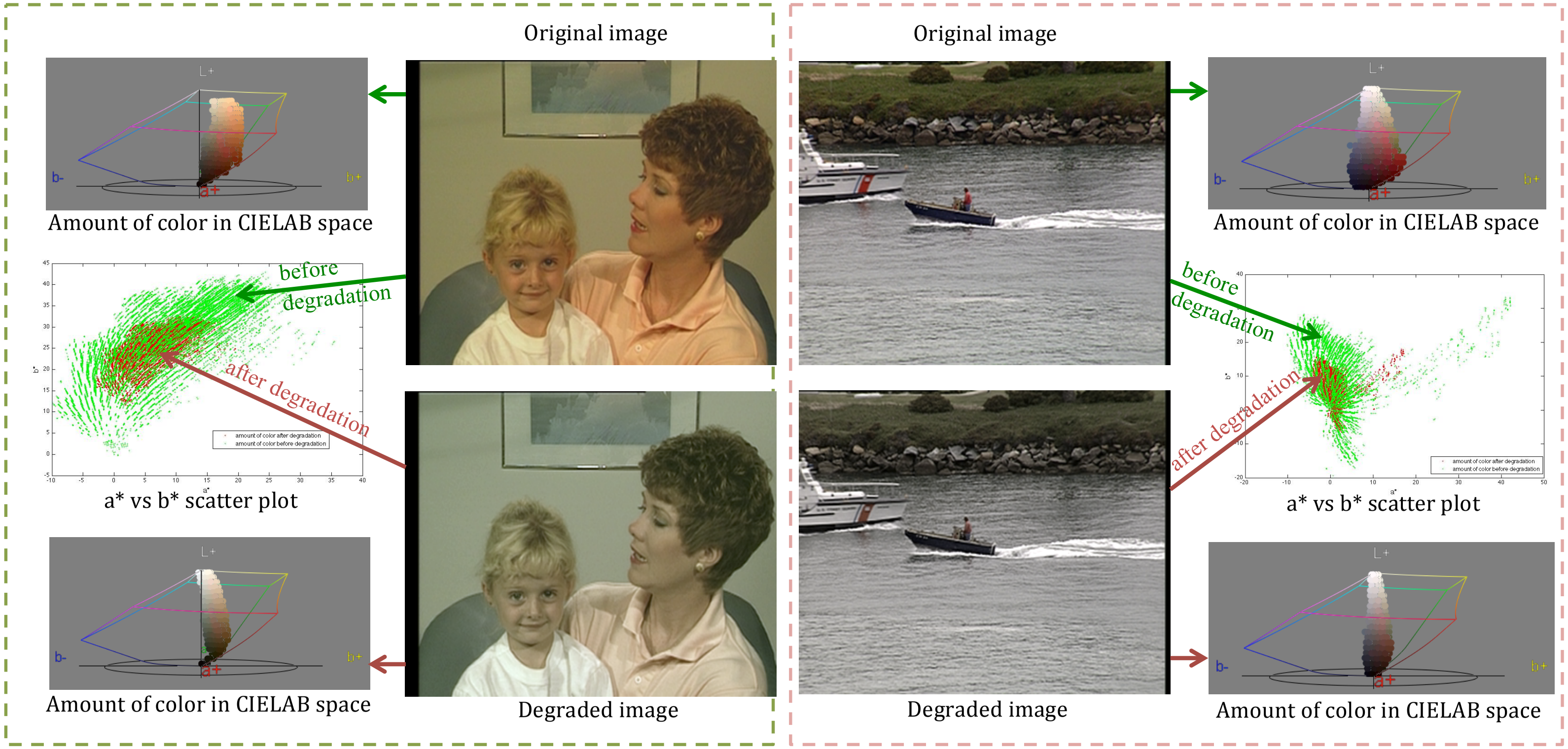

Acquisition site
Receiver site
Non-degraded KFs and NKFs
Cam1
NKF degradation
Degraded NKFs
Non-degraded KF
Encode KFs and degraded NKFs and transmit
Cam2
NKF degradation
Encode KFs and degraded NKFs and transmit
Camn
NKF degradation
Encode KFs and degraded NKFs and transmit
time
Decoded KFs and NKFs
Enhance degraded NKFs by KFs
Pass information of KF to NKFs
Enhance degraded NKFs by KFs
Enhance degraded NKFs by KFs

Figure 5.2 – Color degradation strategy for the frames.

$$
\begin{aligned}
P_{ind_{final}} &= P_{in}E_{vc} \\
P_{ind1_{final}} &= P_{in1}\left(\sqrt{\frac{E_{vl1}}{E_{vl2}}}\right)^{-k} \\
P_{ind2_{final}} &= P_{in2}(2k)^{-1}
\end{aligned}
\tag{5.2}
$$

In Equation 5.2, P_{in1} and P_{in2} = 1st and 2nd independent axes and k = multiplication factor (determined by experimentation and $0 < k < 1$). We set $k = 0.65$ for our experiments. The value of k should be set higher for the sequences with more balanced color information across all the channels and lower for the sequences with less balanced or less color information; because for the first case, there is sufficient color information to be reduced whereas, for the second case, there is not much information to be reduced.

Since P_{in1}, being the 1st PC in the PCA, has more color information than P_{in2}, we multiplied P_{in1} with $\left(\sqrt{\frac{E_{vl1}}{E_{vl2}}}\right)^{-k}$ and divided P_{in2} with a factor of $2k$. Because $(2k)^{-1}$ is always greater than $\left(\sqrt{\frac{E_{vl1}}{E_{vl2}}}\right)^{-k}$, the color information along P_{in1} axis gets more reduced than the color information along P_{in2} and hence, after the reduction, both $P_{ind1_{final}}$ and $P_{ind1_{final}}$ possess an equivalent amount of color information. After the degradation is completed, we bring back the NKFs to RGB space from CIELAB space for further processing. Figure 5.3 shows the amount of reduction in a* and b* channels by scatter plot and CIELAB



(a)

(b)

Figure 5.3 – Amount of color reduction:(a) Mother and daughter MD (b) Coastguard CG. Both the a^* *versus* b^* scatterplots and the CIELAB diagrams for the NKFs of (a) and (b) show that the degraded NKFs contain less color information than the original NKFs.

diagram for two NKF frames from MD and CG. We can perceive the color reduction by comparing the shirt and skin color of *MD* in Figure 5.3(a) and the shirt of the person on the small boat, red strip of the ship and green color of the plant in Figure 5.3(b). Therefore, after the degradation, we get NKFs with reduced color information and size.

For the experimental result comparison in Section 5.3.3, we also devised the *inverse process* of this degradation method in which we multiply P_{in1} with a factor of $2k$ and P_{in2} with $\left(\sqrt{\frac{E_{vl1}}{E_{vl2}}}\right)^k$. Since $\left(\sqrt{\frac{E_{vl1}}{E_{vl2}}}\right)^k$ is always bigger than $2k$, P_{in2} is enhanced more than P_{in1} and so the color is enhancement is balanced for both P_{in1} and P_{in2}. In the next section, we discuss about our two proposed methods which we use to enhance these degraded NKFs after they are decoded on the receiver site.

5.3.2 Enhancement process

After the NKFs are degraded at the acquisition site and encoded with non-degraded KFs and transmitted to the receiver site, these NKFs and KFs are decoded and then the NKFs are enhanced with the information stored in the KFs. An *inverse* operation of the degradation process (described in previous section) doesn't help to enhance the color information of the NKFs, because during the degradation that color information is already lost and the *inverse* operation, without any other enhancement process, cannot help to elevate the color information of NKFs to the KF's level. We propose two enhancement methods by which the degraded NKFs level up to the color information of the non-degraded KFs.

At the beginning phase of enhancement, the mean values of each channel (RGB) of the KF are stored and these values are used to level up the color information of NKFs. Each of the channels of the NKFs is scanned individually and mean values of corresponding channels of KFs are passed to the NKFs to level up that channel's color information. For the calculation, we used the channel's color value range from 0 to 1, later we multiplied the channel values with the original ranges to get back to the original values. The diagram in Figure 5.4 shows the steps of our proposed methods. Details about the methods are explained in the following subsections.

5.3.2.1 NKF enhancement: method 1

In this method, we elevate the color information of each of the channels (RGB) of the NKFs by using the mean values of RGB channels of KF which were stored initially. We use the formula in Equation 5.3 to enhance the NKFs.

$$C_i^E = \begin{cases} \frac{\left(1-\overline{C}_i^k\right)}{\left(1-\overline{C}_i^{nk}\right)} \left(C_i^{nk} - \overline{C}_i^{nk}\right) + \overline{C}_i^k, & if\ \overline{C}_i^k > \overline{C}_i^{nk} \\ C_i^{nk} \overline{C}_i^k \left(\overline{C}_i^{nk}\right)^{-1} & otherwise \end{cases} \tag{5.3}$$

In Equation 5.3, C_i^E = enhanced color value for channel C_i, C_i^{nk} = color value of degraded NKF for channel C_i, $\overline{C}_i^k$ = mean color value for the KF for channel C_i and $\overline{C}_i^{nk}$ = mean color value for the NKF for channel C_i. Here, if the mean color value $\overline{C}_i^k$ is greater than the mean color value $\overline{C}_i^{nk}$, then all the color values of that channel are uplifted similar to the top illustration of diagram (i) of Figure 5.4, but if $\overline{C}_i^k < \overline{C}_i^{nk}$ (which could occur very seldom), then the channel values of NKFs follow the mean value of the channels of KF; see the bottom illustration of diagram (i) of Figure 5.4. In this way, the information stored in the KFs are used to elevate the color information of the NKFs.

5.3.2.2 NKF enhancement: method 2

Here, we use the following strategy to enhance the NKFs: we calculate the ratio of mean color value for each channel of NKF and KF, and then take the maximum from these three ratios and inverse multiply the maximum ratio with the ratio of mean color value for each channel of KF and mean color value for each channel of NKF($\frac{\overline{C}_i^k}{\overline{C}_i^{nk}}$), and with the color value for each channel (C_i^{nk}) of NKF. We use the formula in Equation 5.4 for this process.

$$C_i^E = C_i^{nk} \frac{\overline{C}_i^k}{\overline{C}_i^{nk}} \left(max \left\{ \frac{\overline{C}_1^k}{\overline{C}_1^{nk}}, \frac{\overline{C}_2^k}{\overline{C}_2^{nk}}, \frac{\overline{C}_3^k}{\overline{C}_3^{nk}} \right\} \right)^{-1} \tag{5.4}$$

In Equation 5.4, C_i^E, C_i^{nk}, $\overline{C}_i^k$ and $\overline{C}_i^{nk}$ represents same as in Equation 5.4 and $\overline{C}_1^k$, $\overline{C}_2^k$, $\overline{C}_3^k$, $\overline{C}_1^{nk}$, $\overline{C}_2^{nk}$, $\overline{C}_3^{nk}$ represents the mean color values for the red, green and blue channels of the KF and NKF respectively. The diagram (ii) in Figure 5.4 illustrates the enhancement method using this approach.

5.3.3 Experimental results

For evaluating our proposed method, we used three reference data sets from [112] among which two are CIF (352 × 288) video sequences from *Mother and daughter* MD and *Coastguard* CG datasets and another is 4CIF (704 × 576) video sequence from *ice* dataset. Here, we used *FFMPEG*'s MPEG-4/AVC encoder (mentioned as *reference coder* in this section) to encode the stream at low bitrates. Since, live stream of frames is transmitted in a normal communication scenario, we used two seconds' worth number of frames to test our methods to avoid noticeable transmission delay. We encoded the stream at 30 fps, so there were 60 frames for each of the tests. We compared the PSNR gain with our degradation and enhancement model and with the output from the reference coder.

We had two test phases. In one test phase, we varied the bitrate from 5 kbps to 35 kbps, kept the quantization parameter QP at 31 and measured the PSNR gain of the NKFs for our proposed methods, reference coder, only decoded situation and decoded & *inverse*

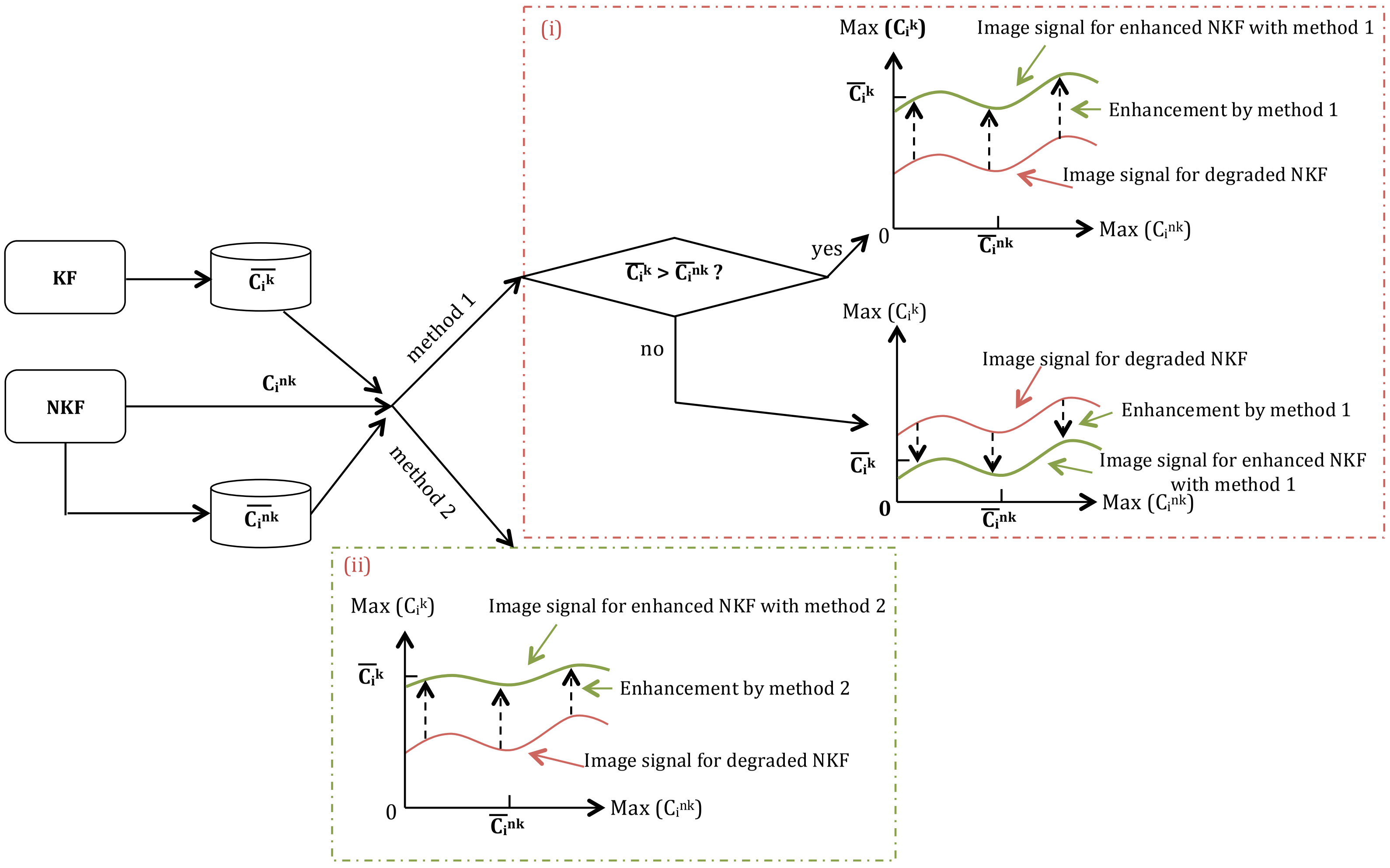
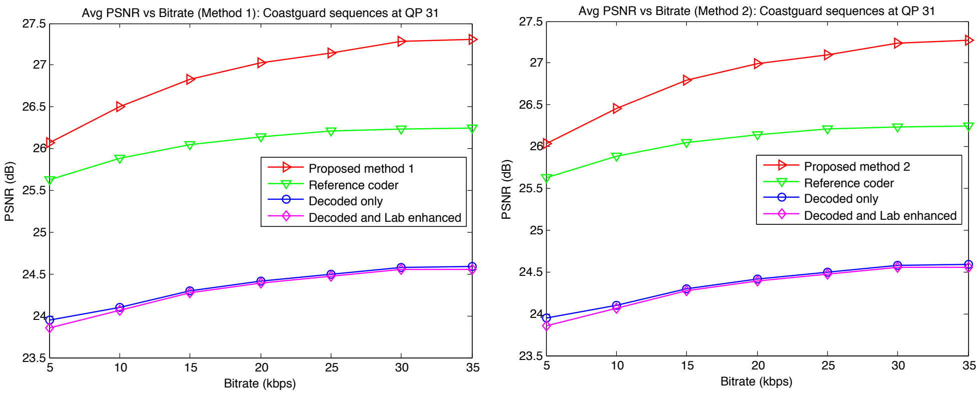


Figure 5.4 – Diagram of proposed enhancement methods.

process of degradation (referred here as *Decoded and Lab enhanced*). In case of *Decoded and Lab enhanced*, after decoding the NKFs at the receiver site, we applied the *inverse process* of the degradation (discussed in Section 5.3.1.1). In the second test phase, we varied the frequency of KFs and considered every *2nd, 5th, 10th, and 20th* frames as KFs respectively for the four cases. Then, we measured the PSNR gain for the NKFs with our proposed methods and reference coder.

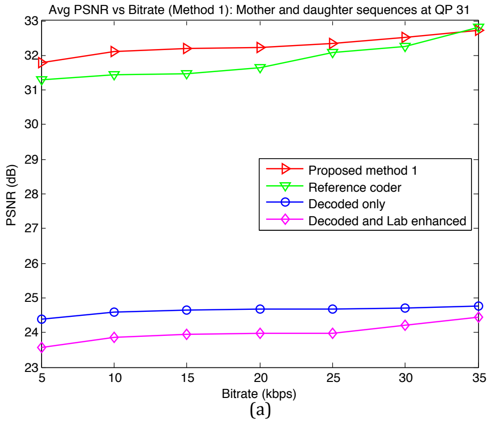
For the first phase of the test, we calculated the average PSNR for four scenarios. In *first scenario*, frames are degraded, encoded, transmitted, decoded and enhanced by our strategies, in *second scenario*, frames are not degraded, they are encoded, transmitted and then decoded (here only reference coder is used), in *third scenario*, frames are degraded, encoded, transmitted and decoded, but they are not enhanced and in *fourth scenario*, frames are degraded, encoded, transmitted, decoded and enhanced by the *inverse process*. The results of these four scenarios are depicted in Figure 5.5 and 5.6.

In Figure 5.5 (a) and (b), we can see that the PSNR gains from the 3rd and 4th scenarios are much lower than the other two scenarios. While the reference coder output, i.e. the 2nd scenario, shows good PSNR level, our methods (the first scenario) show better PSNR level than all the other scenarios. Our results show clear improvement in compression ratio i.e. PSNR gain. We perceive similar superior PSNR gains, as in Figure 5.5, with our proposed strategy for the *MD* and *ice* sequences as well; see the results in Figure 5.6. The PSNR gain for *ice*, in Figure 5.6(b), is comparatively less than the gain of the *MD* and *CG*. It occurs because the *MD* and *CG* sequences contain much balanced color information in all the channels and hence, the color information degradation and enhancement is better and balanced than the *ice* sequence. The PSNR gain for *ice* sequence could be improved by using a lower value for k in Equation 2, however, we chose to use the same value of k for ensuring the uniformity of the experiment settings.



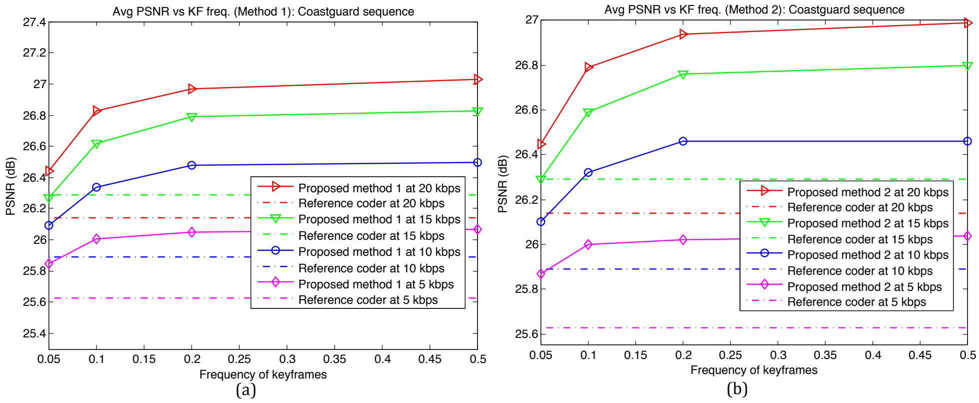
(a) (b)

Figure 5.5 – Enhancement results for Coastguard CG sequence: (a) Proposed method 1, (b) Proposed method 2.



(a) (b)

Figure 5.6 – Enhancement results with proposed method 1 on: (a) Mother and daughter MD sequence, (b) *ice* sequence.

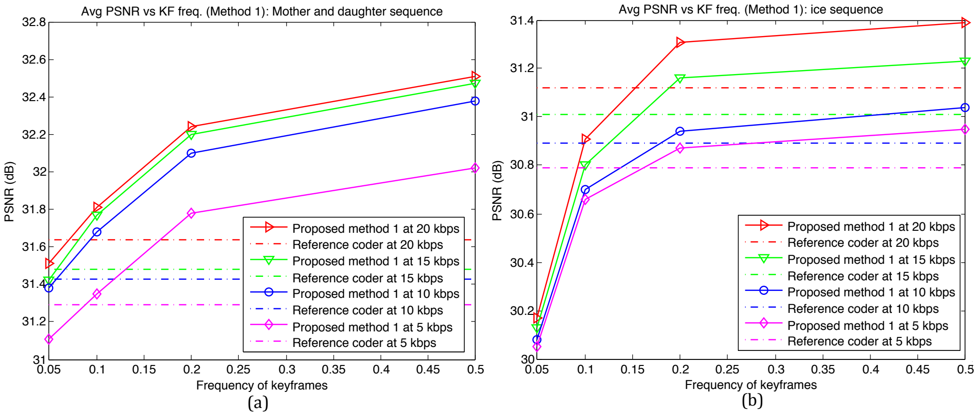


(a) (b)

Figure 5.7 – Enhancement results varying KF frequency for Coastguard CG sequence: (a) Proposed method 1, (b) Proposed method 2.

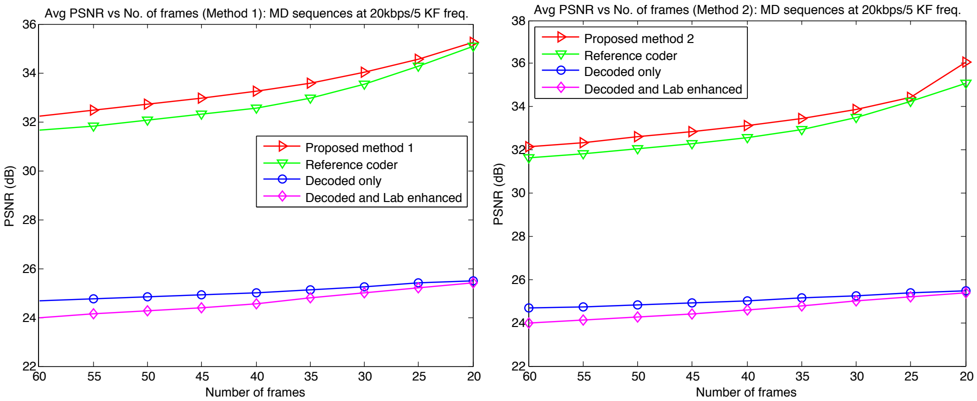
For second phase of the test, we varied the frequency of KFs (from 2 to 20 KF frequency) and displayed the PSNR gain. For this phase, we calculated the average PSNR for the frames for two scenarios. In the *first scenario*, frames are degraded with different KF frequencies, encoded, transmitted, decoded and enhanced by our strategies and in the *second scenario*, frames are not degraded, then encoded, transmitted and then decoded (no enhancement strategy is used).

Figure 5.7 shows the PSNR gains from our proposed methods with varying KF frequencies at different bitrates (1st scenario) and the PSNR gains from the reference coder at different bitrates (2nd scenario). Figure 5.7 shows that our method outperforms the reference coder for all the bitrates for most of the KF frequencies; as the KF frequency decreases from 20 to 2, we can see clear PSNR gain improvement. In the case of *MD* and *ice* also, our strategies show better PSNR gain than the reference coder; results are depicted in Figure 5.8.



(a) (b)

Figure 5.8 – Enhancement results varying KF frequency with proposed method 1 on: (a) MD sequence, (b) *ice* sequence.



(a) (b)

Figure 5.9 – PSNR result for MD sequence: varying total no. of frames.

Figure 5.9 shows the PSNR gain with varying total number of frames from 60 to 20. Both of our methods, in Figure 5.9 (a) and (b), provide better PSNR gain than the reference coder and also the *third* and *fourth* scenarios of the first phase. Our methods perform better than the reference coder because the reference coder produces color artifacts in all the color channels during encoding while our methods successfully enhance those degraded color information with the information stored in the KFs. In Figure 5.9, as the total number of frames decreases, we can see a rise in PSNR gains; it happens because, with fewer frames, the overall artifacts produced by the reference coder become less and hence the PSNR gain is improved. The total number of frames to be used in this case is still under investigation.

We have presented a subjective comparison test in Figure 5.10 for evaluating our proposed methods. In this test, we considered every 5th frame as KFs and used 20 kbps bitrate with 31 QP to process these frames with the reference coder. We perceive a clear

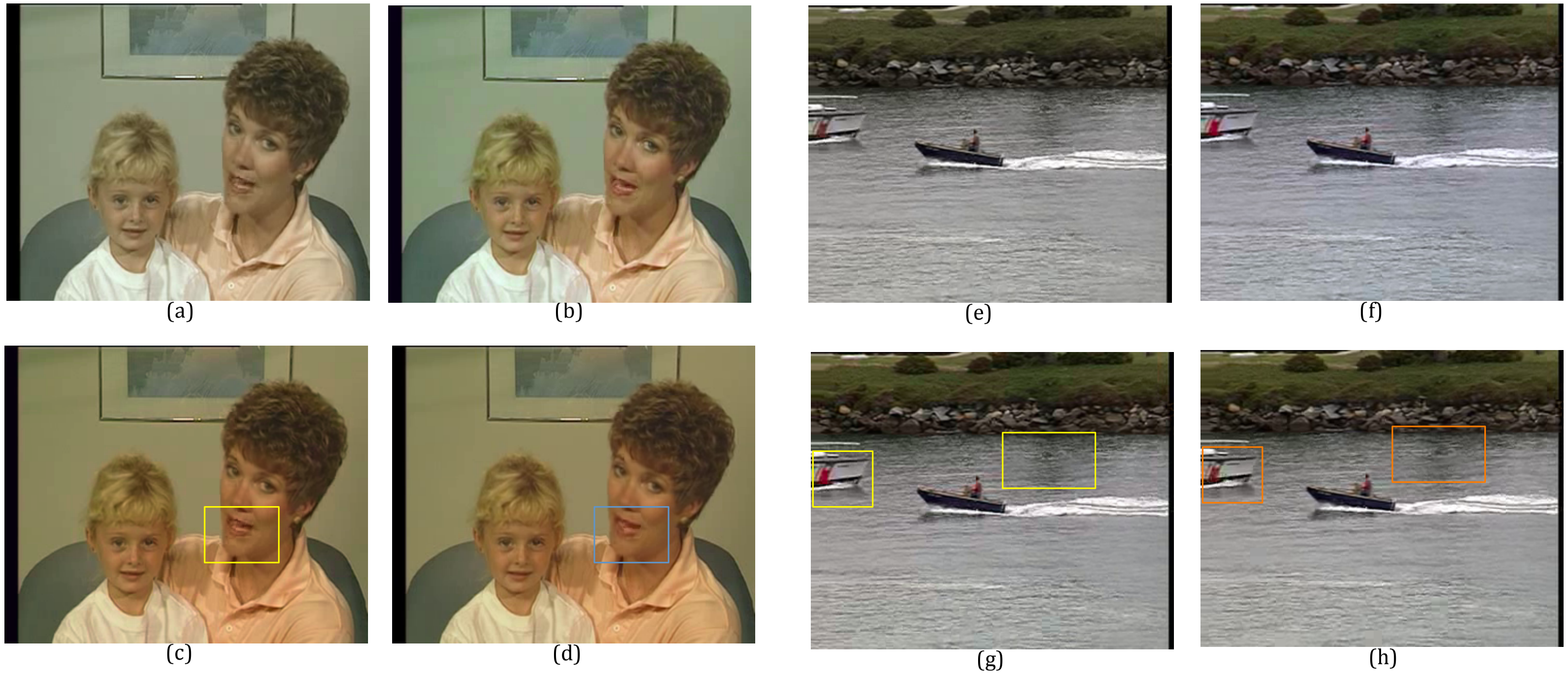


Figure 5.10 – Subjective result comparison for frame 12 of MD and CG datasets: (a,e) decoded only frame of MD and CG, (b,f) *inverse process* of degradation (Lab enhanced; see last paragraph in Section 5.3.1.1) frame of MD and CG, (c,g) enhanced, by our methods, frame of MD and CG, (d,h) non-degraded reference coder frame of MD and CG.

improvement in color quality for the images in Figure 5.10(c) and (g) in comparison with Figure 5.10(b) and (f) respectively. Figure 5.10 shows that the enhanced frames of *MD* and *CG* are visually more pleasant than the reference coder output. The rectangle marked area in Figure 5.10 (c) shows that our method enhances the frames while keeping the details (see the chin, lips and teeth) of the original frame in comparison with the reference coder output marked with rectangle in Figure 5.10 (d). We also perceive similar color enhancement performance from our methods, in Figure 5.10 (g), which preserve more details than the reference coder. Here, on the marked areas, such as the red strip of the ship and the water area, our methods provide better color information with more details than the reference coder output in Figure 5.10 (h). As a result of using our data reduction strategies we were able to transmit more data, previously 3.5 Mbps and now 5.27 Mbps, at a given time on a low-bandwidth network. It is worth to mention that, our proposed approach itself is not a coding scheme, rather it is used as a supporting system to the classical coding schemes in order to elevate the compression rate of the coder.

5.3.4 Discussion, limitations and future work

We have presented a camera data reduction method which improves the encoder performance at the acquisition site. As a result, a large amount of data can potentially be transmitted at low bitrates via a lower-bandwidth network. The degradation method, on the acquisition site, reduces the image data by decreasing the volume of color in the NKFs and keep the KFs intact. On the receiver site, the degraded NKFs are enhanced by using the information from the non-degraded KFs. The PSNR comparison, on three reference sequences, between our methods and the reference coder indicates that our methods gain better PSNR value. The subjective comparison also shows that the outputs from our methods are visually more pleasant than the reference coder output.

Although our proposed approach obtains better PSNR gains for the MD and CG sequences, PSNR gain is not similar for the *ice* sequence which has less balanced color information across all the channels. The limitation is due to the fact that, if the images already have less color information across the channels, there is less room for the degradation and enhancement process and thus the PSNR gain might not be up to the level of the images with more balanced color information. Moreover, if we transmit more than 300 frames at a time, our proposed strategy would perform comparatively slower; because, the decoder produces more color artifacts in the later frames which demands more processing to elevate the color information for those frames. Since we consider transmitting two seconds' worth of frames (i.e. 60 frames), this limitation does not affect our tests.

In the future, we plan to examine our approach in conjunction with existing SR methods for obtaining a better compression ratio. Moreover, we also plan to use our approach to examine the compression rate gain for HD image stream.

5.4 Data reduction using display model and light field geometry

Here, we discuss our second data reduction strategy which deals with the large volume of data captured with multiple cameras that are processed to be displayed on a light filed display. Since using classical coding techniques directly on large data might not guarantee efficient data reduction [97], it would be beneficial to determine the actual portion of data needed by the display system and discard the rest of the input data to ensure real-time transmission. To this end, we propose a fast and efficient data reduction strategy for systems equipped with multiple cameras and light filed display. Our approach automatically isolates the required areas of the incoming images which contribute to the light field reconstruction. We explicitly consider the display model and, captured and reconstructed light field geometry for devising a precise and automatic data selection procedure. The key contribution here is the reduction of the data being transmitted within a communication system, equipped with a light field display, by finding the optimum region of interest from multiple camera images that is used in light field reconstruction.

5.4.1 Data reduction strategy

Light field displays present a scene in 3D space. They do not simply project multiple views in different directions to create a 3D illusion, it requires complex data processing to do so. This primary observation is the basis of our proposed data reduction scheme. Figure 5.11 shows the whole chain of capturing, processing, rendering and displaying of light field content. A single Linux-based acquisition node controls the data capturing part which consists of 27 compact USB cameras. The captured images are streamed directly to the rendering cluster via a Gigabit Ethernet connection. The rendering cluster then drives the optical modules of the display and finally, a holographic screen is used to realize the 3D information in the form of light rays projected by the optical modules. The processing done by application node includes controlling operations such as camera

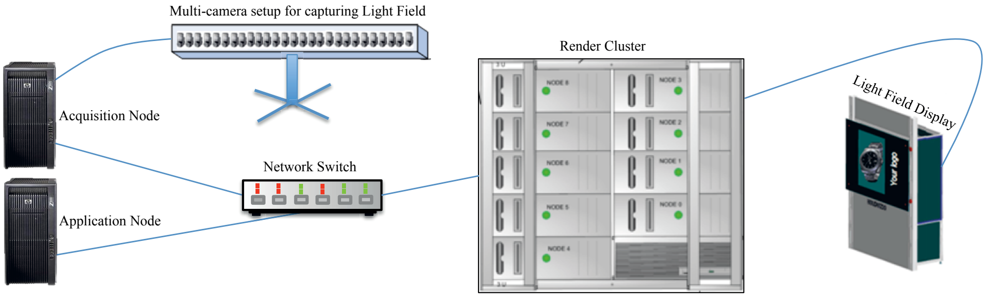
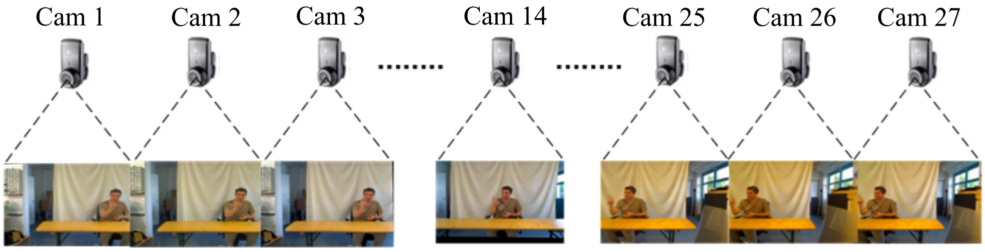
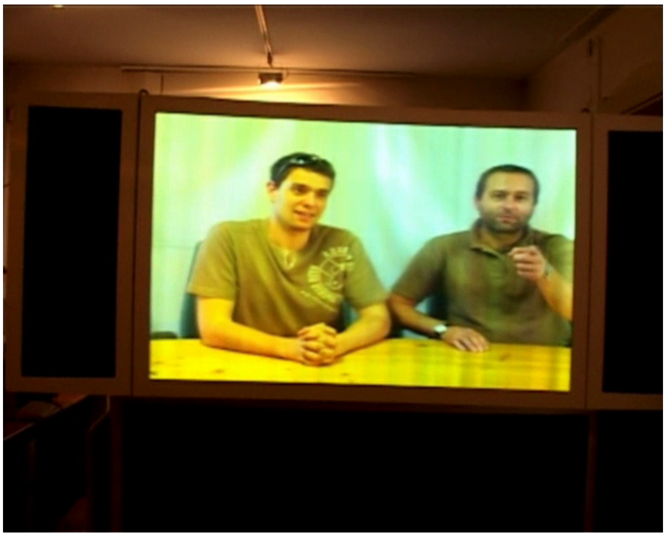


Figure 5.11 – Light field capture, processing and displaying pipeline.



(a)



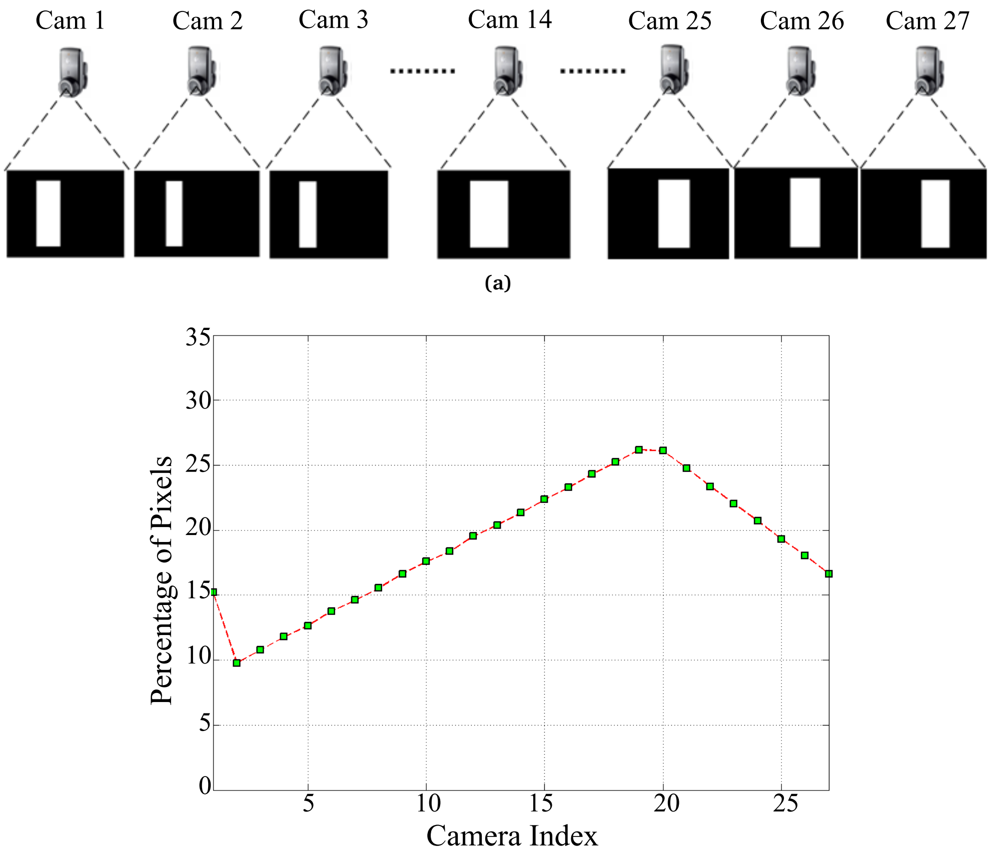
(b)

Figure 5.12 – Sample scene acquisition and reconstruction for light field display: (a) sample light field capturing, (b) light field reconstruction.

calibration and checking the preview from all the cameras. The main part of this processing involves calculating the camera calibration data; a semi-automatic method is adopted for calibrating 27 cameras.

Once the calibration is done, the calibration data is made available to the rendering cluster. The render cluster is equipped with light field modelling data built on display projection geometry beforehand. The incoming pixels of captured image stream are reordered on each cluster node's GPU using the available light field geometry and the camera calibration data. This pixel manipulation is handled using look-up tables, which are specific for each node in the render cluster.

The output of the render cluster is the 3D lighfield reconstruction of the scene obtained from multiple 2D images. Figure 5.12a shows an example light field capture and Figure 5.12b shows the reconstructed light field realized on a light field display. An important observation from the light field reconstruction process is that not all the incoming pixels



(a)

(b)

Figure 5.13 – Proposed data reduction strategy: (a) calculated significant camera image pixels from a sample capture, (b) percentage of pixels from each camera image used in sample light field reconstruction.

are used from all the cameras. Certain regions in each of the camera images are not used during the light field reconstruction. Another important observation is that the look-up tables used for re-ordering the pixels are constructed once in the beginning of the rendering process and remains the same, as far as the mapping between the two light-field remains the same. These key observations form the basis of the proposed method. However, zooming in and out, or shifting the light field mapping forces the recalculation of these tables.

5.4.2 Experimental setup and procedure

For the experiment, we use Holografika's HV721RC light field display which is a large-scale display and can support multiple users simultaneously. We chose this display primarily due to its simplified geometry. As the case with a typical communication system (see

Figure 5.1), e.g., telepresence, we assume that the capturing is done locally and rendering is done at a remote place. Hence, the total data acquisition system is at a local site and the render cluster together with the optical modules and the display are located at a remote place. As this is the first version of such a communication system, we assume that the local and remote site are not far away from each other and communicate via a Gigabit Ethernet connection.

Our goal here is to reduce the amount of data flow without compromising the image quality and we intend to achieve this not by using any coding schemes, but rather considering the display model and camera calibration. In order to achieve this, the first step is to identify the pixels from the input image stream which are discarded after the final rendering. Figure 5.13a shows significant pixel locations (pixels in white) based on the look-up tables in an experimental capture. Here, we used the pixel-to-light ray mapping information to mark the pixels' positions from each of the camera images used by all nodes in the rendering cluster. In Figure 5.13b, we present the percentage of pixels referred to in the look-up tables for pixel re-ordering from each camera image. Note that the asymmetric nature of the curve is the effect of the chosen region of interest (can be observed from Figures 5.12a and 5.12b) and also a part of it is driven by the camera rotation. Moreover, due to the vertical misalignments of the cameras, the top and bottom of the source images are cropped in this mapping between the incoming and outgoing light fields (we essentially zoomed inside the light field), hence the unused pixels on the top and bottom of some images. A higher ratio of used pixels could be achieved in synthetic setups or with a more precise camera system.

One important observation from Figure 5.13b is that the number of pixels utilized from each camera frame is not the same for any two cameras. Moreover, these pixels are not chosen in the same pattern and from the same locations on every image. However, the significant pixels from every image form a rectangular shape of varying area across multiple camera images and once set, the shape and the area remains same throughout the capture. This means, light field from the current capture setup can be comfortably reconstructed using the patterns of useful pixels on multiple camera images. Hence, we can recreate exactly the same 3D impression with suitably chosen pixel subset on camera images using the masks in a pre-calculated pattern. We exploit this observation to reduce the amount of data being transmitted.

At first, we shift the look-up table generation mechanism from the remote site to the local site and include it as a part of the processing on the transmission side. The look-up table generation is carried out before transmitting the data and after finalizing the calibration. These tables are generated for all the nodes and once this is done, we introduce an additional processing step on all the camera images where we create a mask for each camera image that defines a pattern of significant pixels needed by all the nodes in the render cluster. The incoming camera images are carefully tailored using the created masks. As soon as the look-up tables are made available, this step is executed very fast without involving a lot of processing. Thus we can create a one-to-one mapping of the incoming and outgoing camera images. The processed camera images are now light-weight and are sent to the remote site. To speed up the process of creating masks and accessing the image pixels locally, we introduce an additional processing computer on the local site.

Now, since the look-up tables are already available, the rendering cluster does not require additional time generating them. Moreover, since the camera calibration data are also included in the look-up tables, they do not need to be transmitted separately. Thus, instead of sending camera calibration data to the render cluster, we send the lookup tables for each node before the rendering process. The render cluster uses the look-up tables and the reduced image data to perform the light field rendering. As we discard parts of camera images, the output image texture coordinates may not coincide with the coordinates in the look-up tables. Thus, we need to store texture coordinate offset values in both X and Y direction for all the 27 cameras. The 54 valued offset texture is also transmitted before the actual rendering starts.

5.4.3 Results, discussions and future work

We tested the performance of our approach on a pre-recorded 19 second footage, "Telepresence" using a light field display. With the given initial conditions, we demonstrated that our approach yields the same light field reconstruction without introducing any temporal or spatial artifacts and yet using only up to 20% of the whole data stream. Thus, the bandwidth resource consumption is effectively reduced by a factor of five. Also because of the reduced image resolution, GPU uploading and hence the overall rendering at the remote site becomes faster. Although the amount of data being uploaded is reduced, for the final rendering the number of texels used remains the same and thus there is not any significant speed up in the rendering frame rate.

Here, we presented a lossless approach to reduce the data flow in a multi-camera setup using a light field display. Our method does not rely on any coding schemes, but rather uses the display projection geometry to exploit and eliminate redundancy. We proposed minor changes in the capturing, processing and rendering pipeline with additional processing at the local site that helps achieving significant data reduction. Furthermore, the additional processing step before transmission, mostly involves simple image processing operations such as generating masks and extracting a bunch of pixels and needs to be done only once. The processed and transmitted data not only consumes less bandwidth but also speeds up the texture upload process.

Here, we showed the use of global masks to reduce pixel data selectively. In practice, each rendering node does not need the whole information, even from the extracted pixel subset. It is possible to customize the masks for each of the render cluster nodes, which can further improve speed. Also, the camera images can be subjected to a 90 degree rotation soon after the capture and then, we would access the pixels row-wise for selective transmission. This might bypass any unnecessary passages during the memory access and direct memory offsets can be used.

In the proposed method, we made an assumption that the local and remote sites are connected via a low latency, relatively high-bandwidth connection. In general, this is not the case and in order to transmit the light field data over longer distances, it is possible to incorporate multi-view coding schemes such as H.264, MVC, and HEVC. Also, the capturing speed at the acquisition site is a bottleneck in the current setup. Using constant exposure time cameras with hardware trigger might further increase the accuracy in the camera synchronization.

Chapter 6

Use cases of our proposed strategies

Here, we discuss the application areas of our proposed approaches. Firstly, our proposed depth enhancement strategy can be applied to enhance the depth frames in many application areas including but not limited to telepresence, 3D reconstruction of a scene, e-learning, tracking, forensic analysis and alike. Besides this, our data reduction approach can also be applied to reduce the volume of required data which need to be transmitted for generating the output. Below, we discuss briefly how our approaches can be applied in the different application sectors. The concept of multi-camera setup and 3D reconstruction from the camera data in telepresence and the idea of e-learning application has been published in our article [113, 114].

6.1 Utilization in telepresence applications

Telepresence, an emerging application area which often needs 3D scene reconstruction, can be benefitted from our proposed depth enhancement strategy. In the case of telepresence, our method can be used to enhance the stream of frames received from the remote site of the communication. There are typically two sides of communication in a traditional telepresence system, which are: acquisition site and receiver site. Users staying at each site can communicate and interact with each other via network channel; their surrounding environment is captured and transmitted to the other end of the communication site. Figure 6.1 illustrates a typical telepresence scenario along with how our depth enhancement strategy can be applied to such a scenario. Since a telepresence system involves processing the captured frames with dynamic objects within them, our method can be suitably applied to enhance those frames which then can be used to obtain an enhanced final output. Besides, since a typical telepresence system is equipped with multiple cameras, our data reduction strategies can also be applied to reduce the volume of generate data and comfortably transmit them over low bandwidth network.

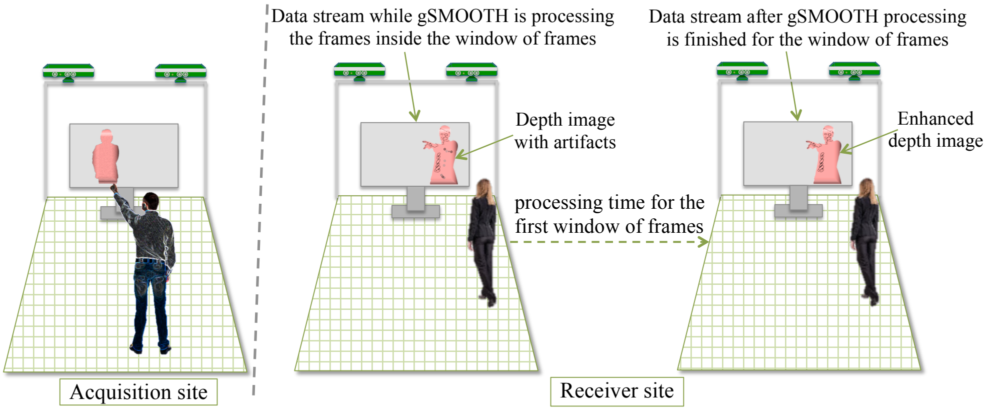

Data stream while gSMOOTH is processing
the frames inside the window of frames
Data stream after gSMOOTH processing
is finished for the window of frames
Depth image
with artifacts
Enhanced
depth image
processing time for the
first window of frames
Acquisition site
Receiver site

Figure 6.1 – Illustration of using our gSMOOTH in a telepresence scenario. The depth image on the right screen of the local site is enhanced by gSMOOTH while the left depth image shows artifacts (since the processing of the frames inside the sliding window is in progress).

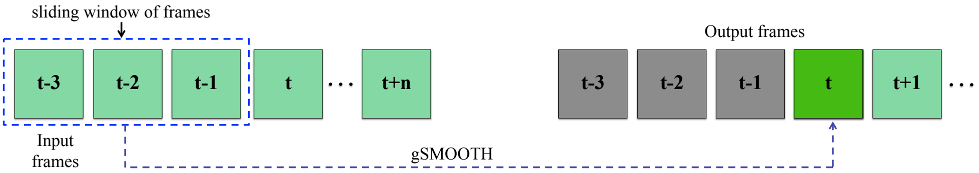


Figure 6.2 – Illustration of sliding window of frames in gSMOOTH. Here, there are three frames inside the sliding window, hence we will perceive the enhanced output from the fourth frame.

Since, in case of our depth enhancement approach gSMOOTH, we use a sliding window of frames to enhance the t_ith frame, we will see the enhanced t_ith frame after processing the frames inside the sliding window, see Figure 6.2. While the frames inside the sliding window are being processed, the users will observe the frames with artifacts, but as soon as the processing is finished and the sliding window moves to the next window of frames, the users will start observing the enhanced stream. Since our approach works in real-time processing the frames in the sliding window take very little time and hence the users observe the frames with artifacts for a very short time. As described in the first paragraph of Section 3.2.2, for the later sliding window, only one frame's worth calculation time is needed since the gSMOOTH calculation for other frames have already been done while processing the frames inside the previous sliding window.

Moreover, since in a typical telepresence system, multiple cameras are used to capture the communication environment, our first data reduction strategy can potentially be used to reduce the large volume of color image data and eventually these reduced data stream can be used to transmit over low bandwidth network and be used to reconstruct a colored 3D representation of the captured scene. Moreover, in case of a telepresence setup with light field display, our second data reduction strategy can be applied to reduce the total data volume by a factor of five which is significant and plays a vital role when the real-time transmission becomes an issue.

Apart from these, our depth enhancement approach gSMOOTH can be applied to other applications as well, such as in 3D reconstruction of a scene, tracking objects and alike, which involves a sequence of frames to be processed before carrying out a certain task. In the following sections, we describe briefly the effect of applying our approach in those applications.

6.2 Utilization in efficient 3D reconstruction of a scene

With the introduction of low-cost cameras, reconstructing a scene in 3D using the images of the scene captured from different viewing angles has become popular in many different application areas in computer vision and computer graphics. While there are different methods which use 2D imagery from multiple stand-alone RGB cameras and stereopsis to reconstruct a scene in 3D, many other works use 3D depth cameras which provide depth information of the scene. There are quite a few works, such as in [59, 113], which use multiple depth cameras, e.g. Kinects, to reconstruct a captured scene so that it can be viewed from an arbitrary viewing angle. As we discussed earlier that, such depth camera images are affected by different artifacts which eventually affect the final 3D reconstruction. We can use our proposed gSMOOTH depth enhancement method to remove those artifacts. Since our approach removes those artifacts and generate output in real-time, our approach has the potential to support an interactive speed for reconstructing a 3D scene using processed and enhanced images from multiple cameras. We demonstrate here such a concept on an existing work done by us in [113]. In the original work, we use a simple median filter to remove the noise from the images captured from two Kinect V1 sensors and later represented the scene by blending the frames form these cameras using a dynamic proxy approach [113]. We used our proposed approach to enhance those depth images and the final output improves the 3D representation than the original work. Figure 6.3 shows a comparison between the 3D representations using a median filter used originally in [113] and our proposed depth enhancement method from [83].

For evaluating the performance of our approach on the 3D reconstruction using images from Kinect V2 sensors, we performed another test and found out that our method also efficiently enhances the images from those Kinect V2 sensors and hence the final 3D reconstruction has better surface reconstruction than the 3D reconstruction with original raw depth frames. For this test, we captured the scene with two Kinect V2 sensors and then enhanced the scene with our approach and then reconstructed the scene in 3D using the workflow stated in [115]. Figure 6.4 shows a reconstructed 3D scene of a person standing where the final output shows clear improvement on the surface reconstruction. Another test on a different dataset also shows similar improvement, on the surface of

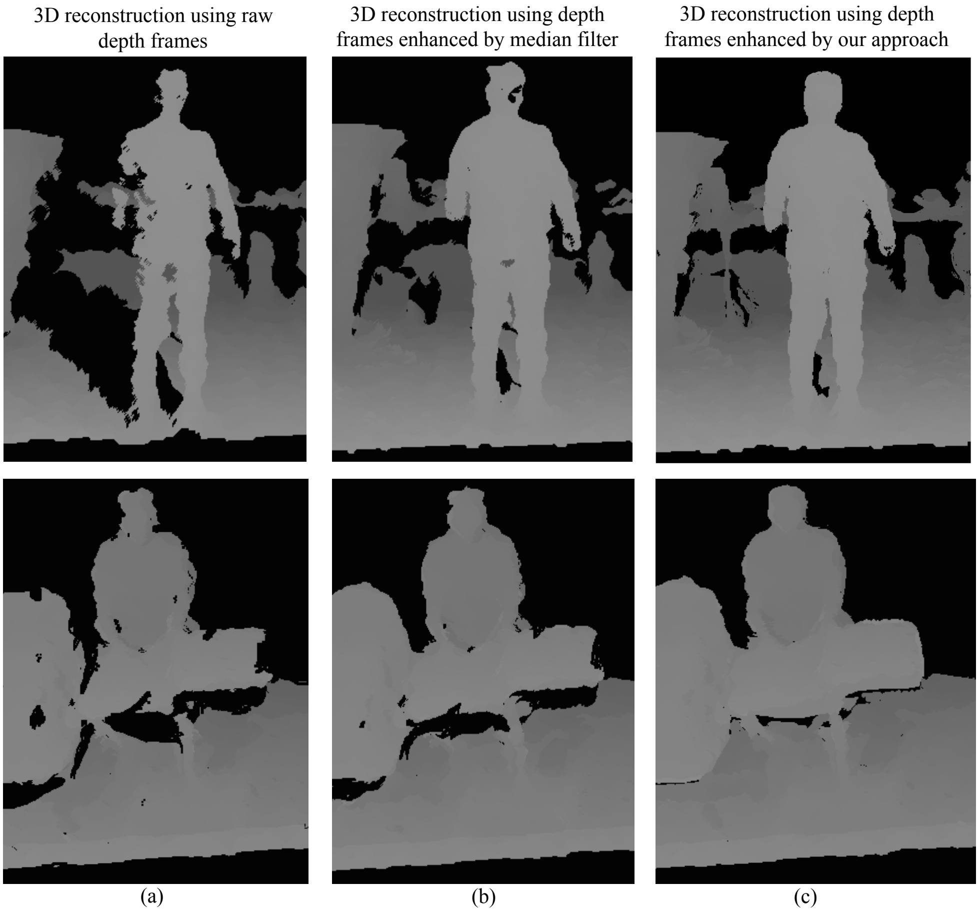

3D reconstruction using raw depth frames
3D reconstruction using depth frames enhanced by median filter
3D reconstruction using depth frames enhanced by our approach
(a)
(b)
(c)

Figure 6.3 – Effect of applying our depth enhancement approach on 3D reconstruction of two scenes. The images on the top row are from the first scene where a person is standing and the images on the bottom row are from the second scene where a person is sitting on a desk. Here, the 3D reconstruction in (a) is using the raw depth frames to reconstruct the scene, while the 3D reconstructions in (b) and (c) use a median filter and our proposed depth enhancement approach respectively. The images in (c) show clear improvement on different parts of the scene, such as the body of the person and the floor areas, in comparison with the images in (b) where a simple median filter is used to enhance the depth frames.

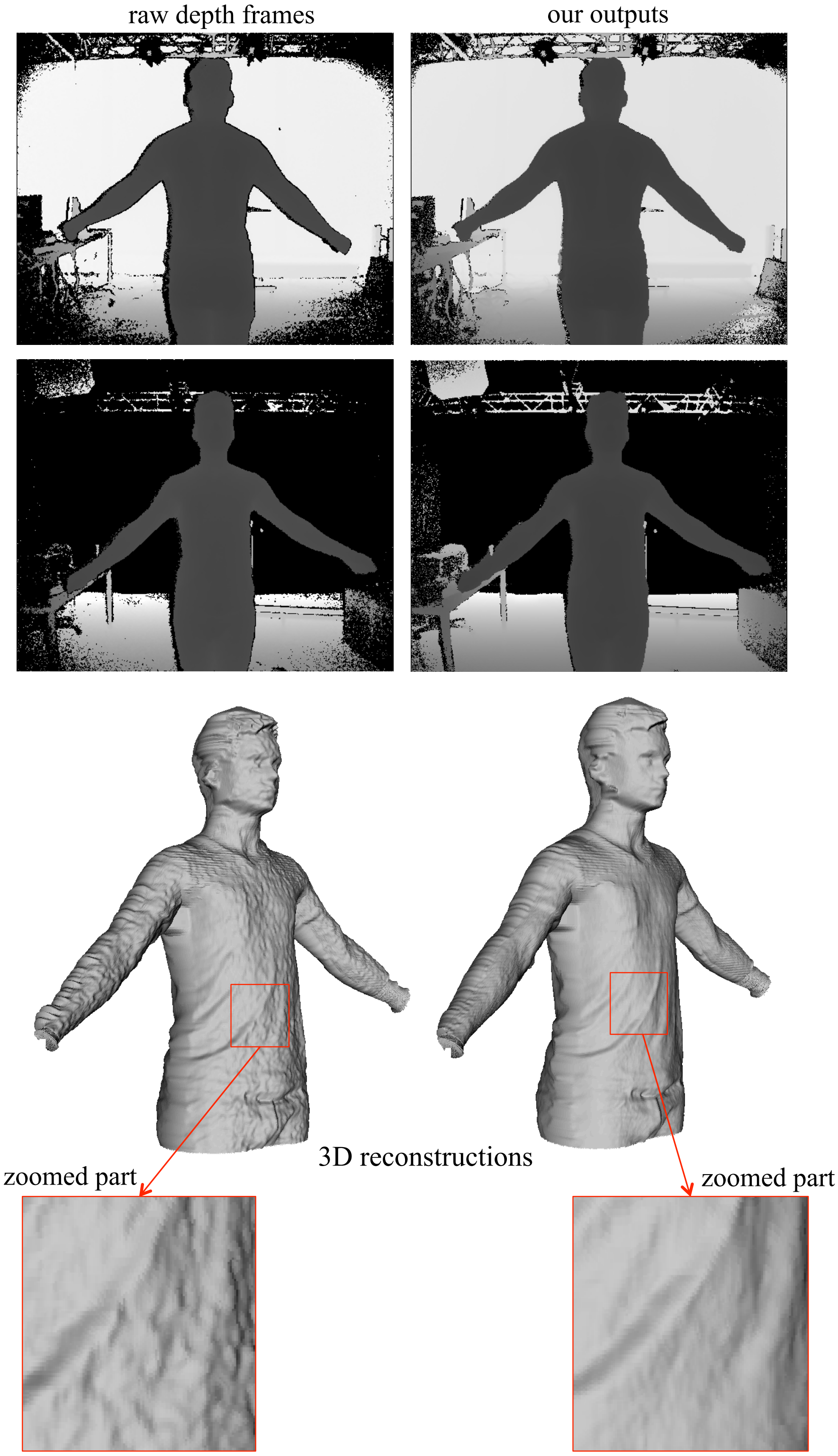
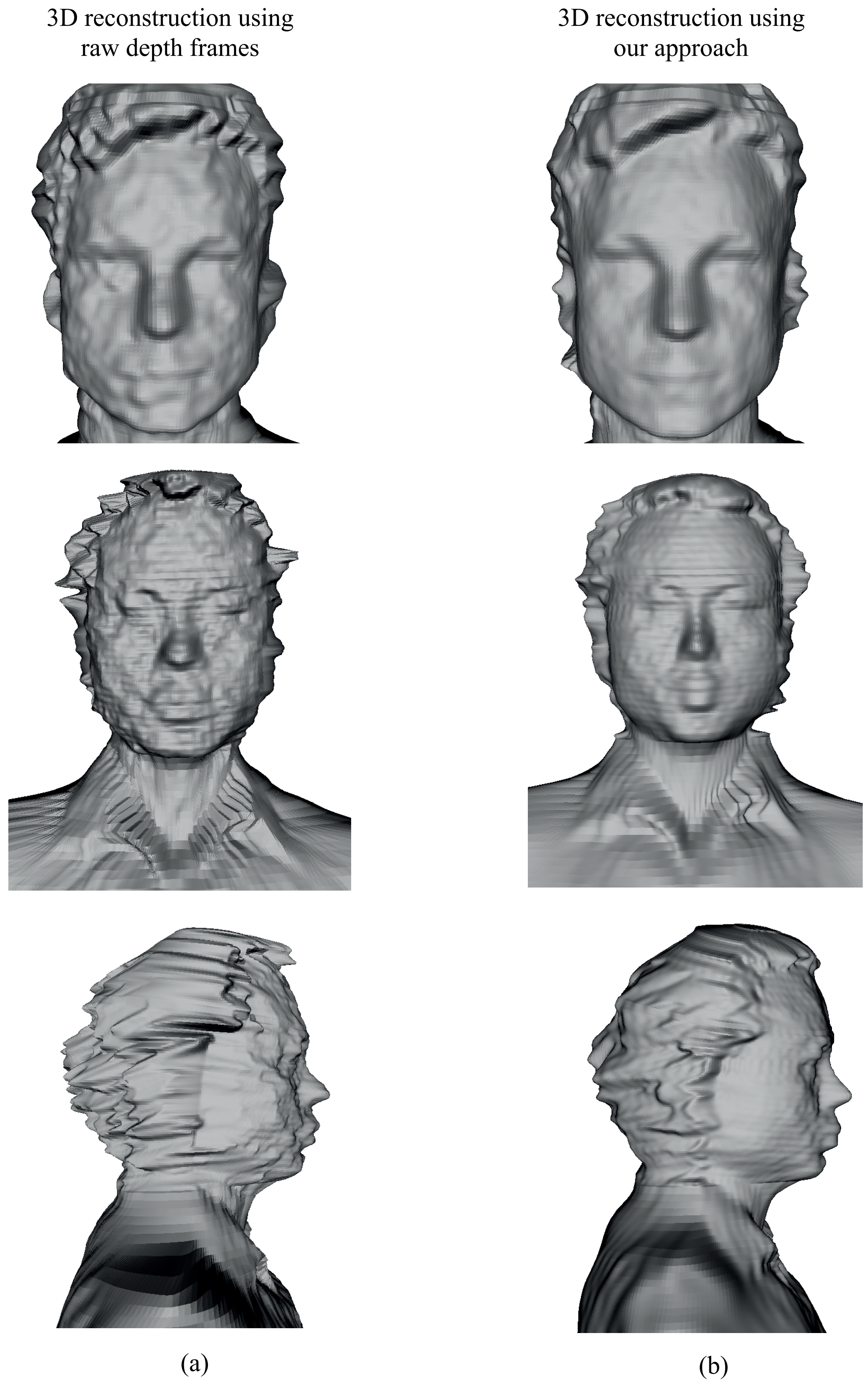


Figure 6.4 – Our depth enhancement result on 3D reconstruction: the raw depth frames on top images shows holes while our outputs removes the hole significantly. The enhancement is reflected on the 3D reconstructed images where the 3D reconstruction on the right side shows improved surface than the 3D reconstruction on the left side. The respective zoomed parts show the detailed enhancement on the surface of the 3D reconstruction.


3D reconstruction using raw depth frames
3D reconstruction using our approach

(a) (b)

Figure 6.5 – Effect of applying our depth enhancement approach on 3D reconstruction of a person's upper body part. Here, the surface reconstruction in (b) using the enhanced depth frames by our approach shows significant improvement in comparison with the surface reconstruction in (a) which uses the raw depth frames.

3D reconstruction, achieved by using our approach; see Figure 6.5. Therefore, we can potentially use our depth enhancement method to a 3D reproduction process and achieve a high-quality output.

6.3 Utilization in e-learning applications

Over the past few decades, the interest in online collaboration on classroom lecture content has been growing steadily [114]. This is happening primarily due to the prospect of online collaboration in enhancing the overall learning experience. Participants of a course are now able to take part in a lecture by being at a different place, through e-learning platforms, than the actual lecture location. They can also access the lecture contents after the lecture is finished. Typically, within an e-learning platform, the lectures are recorded while the actual lecture takes place. Apart from the actual content displayed on a screen via a projector or written by the teacher on a black/white board, the motion of the teacher is tracked by different sensors so that the recorded video can later be classified accordingly [116–118]. An illustration of a classroom where the lecture is being recorded and the lecture content is being displayed by a projector is shown in Figure 6.6.

Typically, when no automatic tracking mechanism is used, then a person usually controls the recording camera to capture different movements of the teacher and the displayed content, and then categorize the entire lecture content accordingly. However recent works, such as in [116–119] suggest that low-cost depth sensors such as a Kinect can potentially be used to capture the lecture content. Since Kinect has the capability to track objects inside a scene and its movement (pan, tilt) can also be controlled automatically, the researchers have started to use such depth cameras to record the lecture contents.

While using a Kinect for recording a lecture, the teacher can use certain gestures for certain tasks. Such as when the teacher wants to start/stop the recording, s/he can make a particular gesture with his hand and then the Kinect, using the embedded tracking method, can act accordingly. Moreover, certain other tasks such as recording the lecture slides displayed on a projector curtain, recording the black/white board on which the teacher has wrote something to explain certain things or record the teacher himself/herself can also be performed with different gestures during the lecture [118]. To achieve these i.e. to execute these actions based on the gestures from the teacher, an accurate recognition of the hand is important.

However, as we know that these depth sensors yield artifacts on the output, hence, consequently these artifacts affect the tracking performance. Due to the presence of these artifacts, objects such as a teacher on a podium inside a lecture room, cannot be tracked with high accuracy. When gesture-based commands are utilized to start and stop the recording of a lecture [118], accurate tracking of the finger becomes crucial. If there are artifacts around the finger location of the teacher, then the tracking lacks

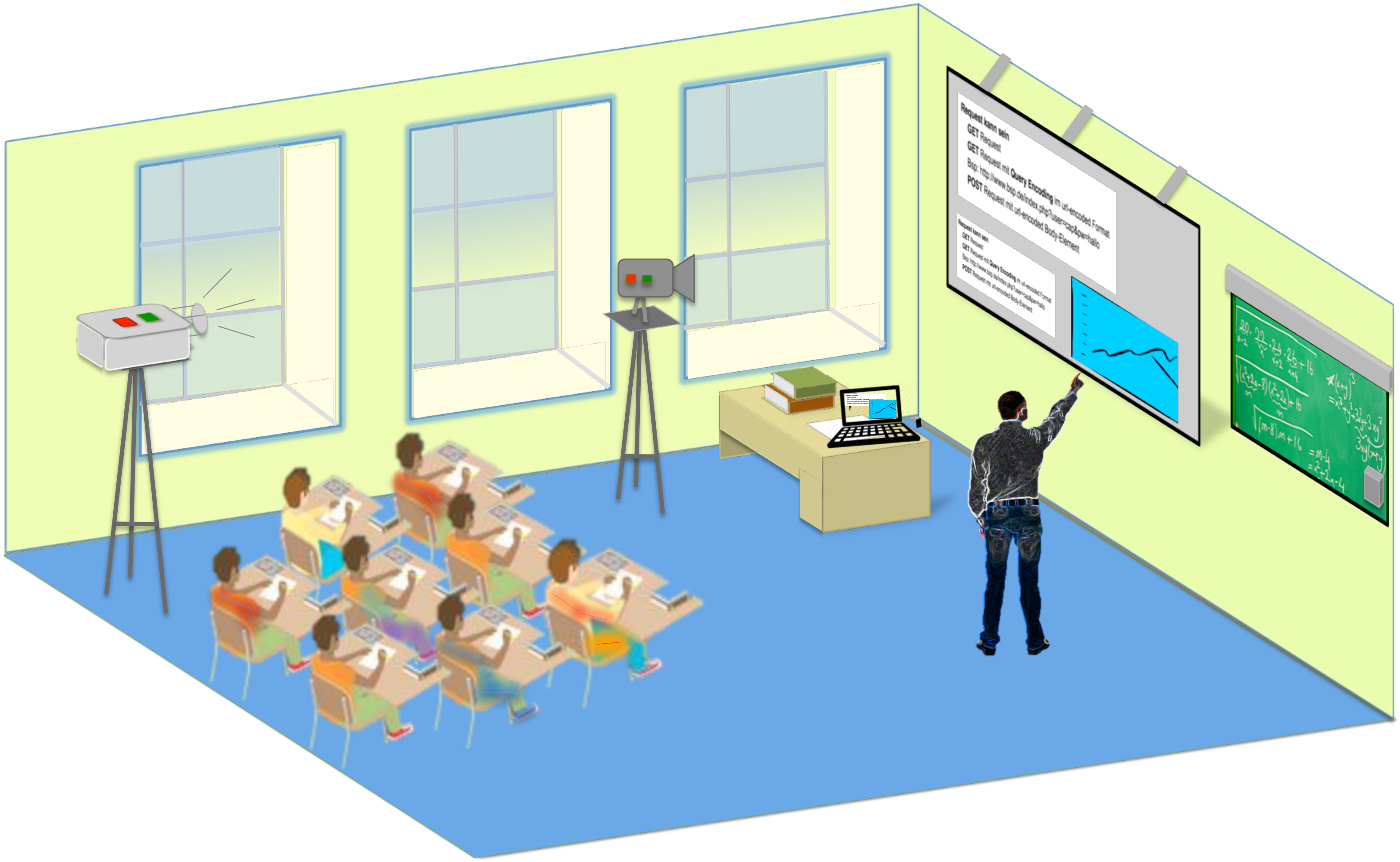

Request kann sein
GET Request
GET Request mit Query Encoding im url-encoded Format
Bsp: http://www.bsp.de/index.php?user=cap&pw=hallo
POST Request mit url-encoded Body-Element
Request kann sein
GET Request
GET Request mit Query Encoding im url-encoded Format
Bsp: http://www.bsp.de/index.php?user=cap&pw=hallo
POST Request mit url-encoded Body-Element

Figure 6.6 – Illustration of recording a lecture for an e-learning system. The lecture is being recorded by camera and the content is being displayed on screen using a projector.

accuracy [120, 121] and hence the recorded lecture will be falsely indexed or categorized. The image in Figure 6.7(a) shows a typical depth frame captured with a Kinect's depth sensor where the frame shows quite a few holes all over the scene. If we look at the hand part of the person, we perceive quite a lot of holes there too. With such artifacts on the detected hand, see the bottom image of Figure 6.7(a), the gesture recognition accuracy would normally be quite low.

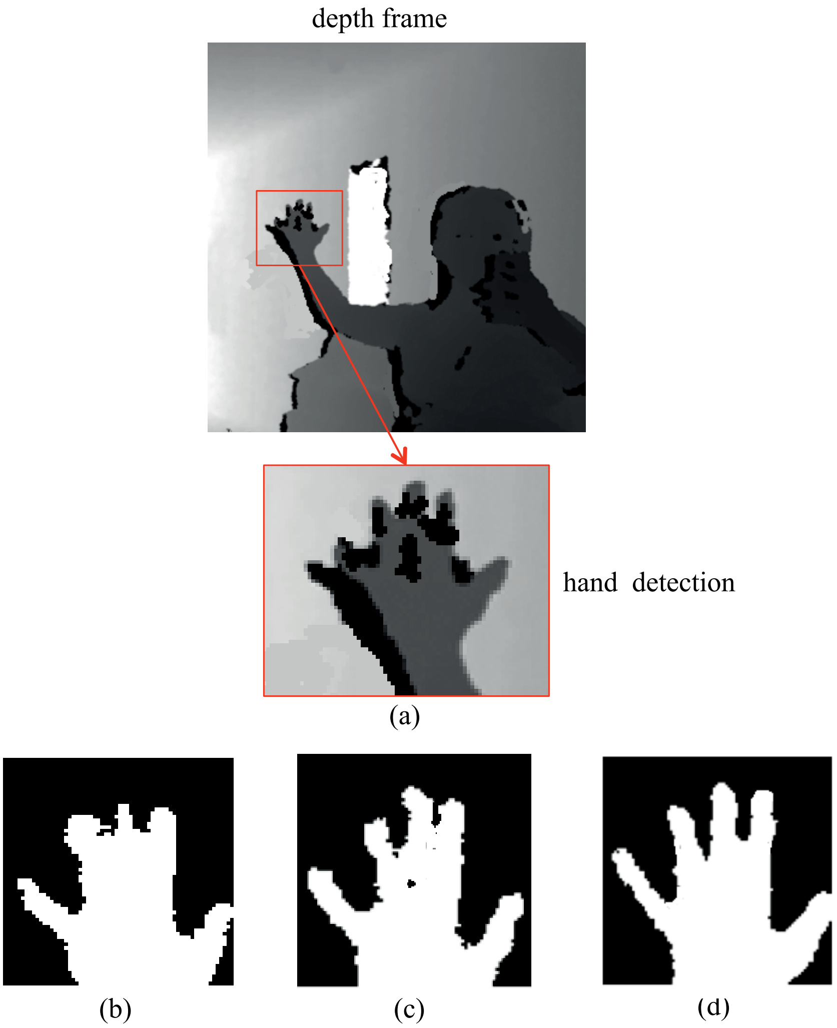


Figure 6.7 – Hand gesture detection and enhancement within a lecture recording session in an e-learning system. Here, a random frame in (a) shows that there are quite a lot of holes appearing in different parts of the captured scene. The detected hand in the bottom image of (a) also shows a lot of holes on it which potentially would affect the performance of gesture recognition. Enhancement methods, such as morphological closing (b) and interpolation technique (c) remove these artifacts partially, however, these outputs do not preserve the homogeneity of the edges of the fingers. Our method in (d) is able to remove a major part of the holes.

After detecting the hand, if traditional smoothing methods, which do not consider to maintain the sharpness of the object's edges, are used for removing the artifacts, they would not preserve the sharpness of the edges of detected hand. Hence, the exact position of the fingers might not be known if such smoothing techniques are applied to remove these noise; see the results in Figure 6.7(b) and (c). Our proposed depth enhancement method can potentially be used to enhance these captured frames. Since our method preserves the sharpness of the edges while enhancing the images, see the result in Figure 6.7(d), the tracking accuracy would potentially be improved. Consequently, with the enhanced frames, the lectures can be recorded and categorized as instructed by the teacher. Hence, our method could potentially be applied in a lecture recording session of an e-learning platform which uses Kinect or other type of depth camera as recording and gesture recognition device.

Chapter 7

Conclusion and future work

In this thesis, our primary focus was to enhance the quality of the depth images captured by the low-cost depth cameras; namely to reduce the artifacts from the resulting depth images and subsequently, the secondary focus was to reduce the amount of captured data for their smooth transmission over low bandwidth network. To that end, we have first introduced a new depth image enhancement framework that fuses both the spatial and temporal domain information of the depth pixels to remove the artifacts from the depth images. We have also developed a ground truth data generation approach which have the potential to further optimize our depth enhancement framework. And secondly, we propose two data reduction strategies which can reduce the amount of camera data needed to transmit to the final processing location/device over the low-bandwidth network.

Our proposed depth enhancement strategy, in Chapter 3, can efficiently remove artifacts from both static and dynamic scenes. Moreover, we are able to achieve real-time processing speed which is crucial for many computer vision applications. Comparing with existing methods in depth image enhancement, our method outperforms most of these methods in case of quality of the final output and in case of processing speed. We applied our depth enhancement strategy to several reference depth sequences and self-recorded sequences to assess the performance of our approach against reference methods. Our method removes the holes, flickering, and ghosting artifacts significantly while preserving the sharpness of the objects' edges. We also used an efficient memory management strategy on the GPU which makes our approach suitable for many real-time applications such as telepresence, e-learning, autonomous driving and alike. As the computational complexity of our method is low and its implementation is straightforward, it can be considered to also address further issues of other RGB-D sensors such as flying pixels with ToF cameras.

From the depth enhancement results, we perceive that there still remains a few holes in case there is no valid depth value for certain pixels in the sliding window of frames. In the future, we would like to remove the remaining holes by trying out different size combinations for the sliding window and the spatial neighborhood window. We would also like to use the knowledge obtained, from Chapter 4, about the noise characteristics of a depth sensor to dynamically determine the optimal window size for a given scene and consequently, improve the optimization of the depth enhancement pipeline. Using the hardware setup and calibration process stated in Chapter 4, we would also like to analyze the noise characteristics of other depth sensors and apply our depth enhancement pipeline on those sensors depth data. To fill the remaining holes, we would also use the depth-hole-filling methods that fill the holes by taking a weighted value of the neighborhood. However, since those methods usually do not consider the homogeneity of the scene-objects and consequently, results in additional artifacts, we have to be cautious about it.

Regarding our data reduction strategies (from Chapter 5), in case of our first strategy of reducing color image data, we have presented a degradation and enhancement model for the color images captured by a multi-camera setup, by which we can improve the performance of the encoder at the acquisition site and send large amount of data at low bitrates via lower-bandwidth network. The degradation method reduces the image data by decreasing the quantity of color data in the non-key frames NKFs and keep the key-frames KFs intact. On the receiver site, the degraded NKFs are enhanced by using the information from the KFs. We used three reference video sequences to compare the PSNR gain between our methods and the reference coder; the results indicate that our methods gain better PSNR value. The subjective comparison also shows that our output is more pleasant than the non-degraded decoded NKFs. In the case of our second strategy, we presented a lossless approach to reduce the data flow in multi-camera telepresence systems using light field displays. The proposed method does not rely on image/video coding schemes, but rather uses the display projection geometry to exploit and eliminate redundancy. We proposed minor changes in the capturing, processing and rendering pipeline with additional processing at the local site that helps achieving significant data reduction. Furthermore, this additional processing step mostly involves simple and light-weight image processing operations which need to be done only once. The processed and transmitted data not only consumes less bandwidth but also speeds up the texture upload.

Although our proposed first data reduction strategy helps to achieve better performance for the reference videos, its performance might degrade due to lack of sufficient color data in all the three channels of a color image. In the future, we plan to test our method in conjunction with the existing SR methods for obtaining a better compression ratio. Moreover, we would also examine the compression rate gain of our method for HD streams. For our second data reduction strategy, since each rendering node does not need the whole information, we would customize the masks for each render cluster nodes which can further improve speed. Besides this, we would incorporate multi-view coding schemes such as H.264, MVC, and HEVC to transmit the light field data over long distances. Furthermore, in order to resolve the camera speed bottleneck issue at the acquisition site, we would like to use hardware-triggered constant exposure time cameras to increase the camera synchronization accuracy and the capturing speed.

List of Acronyms

LMS	Least Median of Squares
NKF	Non Key Frame
KF	Key Frame
SR	Super Resolution
MD	Mother and daughter
CG	Coastguard
LHRD	Large High Resolution Display
PCA	Principle Component Analysis
PC	Principle Component
ToF	Time-of-Flight

List of Figures

1.1 Pipeline of typical depth image enhancement process and our contributions. 10

2.1 Diagram showing the basic differences between Passive and Active depth acquisition techniques. 16

2.2 Illustration of Least Median of Squares: the green dots represent the inliers or valid values in the dataset, the red dots represent the outliers and the thin blue line represents the Least Median of Squares regression line which goes through the inliers. 19

2.3 Diagram illustrating the input data and information domain requirements for the existing depth image enhancement methods. 23

3.1 Invalid and unstable depth values from a depth sensor on successive frames: (a) three successive depth frames, (b) the depth values along a single scan line of the rectangle area in (a), (c) a single depth frame and, the respective output (d) from method [47] showing some ghosting artifacts. 34

3.2 RGB-D frames from Kinect V1 with static objects: (a) color image, (b) four consecutive depth frames of square marked area of (a). The depth values of one pixel inside the diamond marked area in (b) are depicted below the frames. We observe three types of depth value distortions: 2047 is an invalid value (reported occasionally), 923 is an outlier and the others are regular depth values fluctuating because of Gaussian noise. 34

3.3 Illustration of the processing pipeline of gSMOOTH. 36

3.4 Detail processing steps of gSMOOTH for the pixel marked with yellow square. In the spatial processing step, the attributes from the neighbouring pixels of the yellow marked pixel are used to remove the ghosting artifact which occurs on the final output for dynamic scenes. Holes and flickering artifacts are addressed for that pixel in the temporal processing step and finally the enhanced output is generated. 37

3.5 Illustrating the origin of ghosting: The LMS depth enhancement approach [47] presumes a Gaussian distribution of raw depth values in a temporal sliding window for a single pixel (red marked center). Within a continuous depth value change (a), the respective distribution slightly shifts. Hence, the median (blue arrow) also shifts continuously and only valid depth values are finally selected for the enhancement. But in an abrupt change (b), the distribution of the pixels depth value series splits into a foreground and a background distribution. Now, as the median still shifts continuously, intermediate noise values are selected for the enhancement (see rightmost distribution in (b)). Such intermediate values are recognized as ghosting artifacts (green box in (b)) as they do not reflect the abrupt change. 38

3.6 Illustrating of the effect of applying a median filter such as the LMS approach [47] on a window of frames with rather static scene. When we apply the LMS depth enhancement approach on these frames, we can see that the final median values do not vary much from one frame to another and hence, we do not see any ghosting on the final output. 40

3.7 Illustrating of the effect of applying a median filter such as the LMS approach [47] on a window of frames with dynamic scene. When we apply the LMS depth enhancement approach on these frames, we perceive a gradual shift of the median from the lower to the higher pixel value and it stays in between the lower and the higher pixel values. These intermediate values are perceived as ghosting. The bottom sequence of frames shows the ghosting by pointing at the gradient difference, i.e the beginning of the ghosting, between the top and bottom sequence of frames. 41

3.8 Finding edges along the four main directions by calculating the gradients between the green, blue and red pixels average in an exemplary 3-neighborhood. 42

3.9 Depending on the existence and the location of the strongest gradient (dotted line) in a 3-neighborhood $N_{x,y}$, there are 9 different subsets $S_{x,y}$ (darker blues). 43

3.10 Illustrating the principle of our approach. For a current pixel to be enhanced (red marked center) we use our simple gradient-based edge detection to determine that subset (inner orange shape) of the current neighborhood (green rectangle) that is supposed to be most similar to the pixel. The spatial median of such subsets yields a temporal series of candidate values that contains less or no noise. Thus, the distribution of depth values inside the temporal sliding windows is characterized rather by single peaks. Hence, an abrupt change in the raw depth data is reflected by an evolving change in such peaks. Compared to Figure 3.5(b) the median of the temporal distribution (blue arrow) – which is used for the enhancement – now switches abruptly. Finally, the enhanced depth values comprise no intermediate values perceived as ghosting. 44

3.11 Illustration of Spatio-temporal LMS. The temporal window contains a number of frames whose depth values are used to enhance the $(t+1)th$ frame. The spatial neighborhoods **S** are shown for one depth pixel of the frames. 45

3.12 Illustration of our gSMOOTH temporal LMS on one depth pixel in a sliding window. The thick gray line shows the real distance of a scene object over time. The checked circles represent depth values d_{t+i-1} obtained in n consecutive frames. The thin black line approximates the gray line by calculating the estimator M i.e. by minimizing the median of the absolute distances to the depth values, visualized by vertical purple lines. The red marked *invalid value* at $i = 3$ and the turquoise marked *outlier* at $i = 4$ are detectable by our temporal LMS. 46

3.13 Illustration of gSMOOTH temporal LMS on one depth pixel – the values on the top green box are the raw depth values obtained for one pixel over the frames inside the sliding window. 47

3.14 Data management for shared memory. 49

3.15 Performance of gSMOOTH on *PersonBox* dataset (1st row) and on *Chairs* dataset (2nd row): (a) raw depth frames, (b) our result with 3 × 3 spatial neighborhood, (c) our result with 5 × 5 spatial neighborhood. 50

3.16 Performance comparison of gSMOOTH and Nguyen et al. [86] on the *PersonBox* dataset for removing noise and flickering: the top left image is a raw depth frame, the middle image is the output from [86] and the right image is from gSMOOTH. The plot shows the performance of gSMOOTH and method [86] over 15 enhanced frames. The depth values on the plot are the recovered depth values for one pixel in the circle-marked area on the depth frames. gSMOOTH significantly reduces the noise and flickering as seen by the blue line on the plot. 51

3.17 Performance of gSMOOTH on *OfficeRoom* dataset (1st row), *PersonWalking* dataset (2nd row) and *PersonSitting* dataset (3rd row): (a) raw depth frames, (b) results from Nguyen et al. [86], (c) results from Wang et al. [36], (d) our results. For both datasets, our method yields nicer edges while reducing the artifacts. 53

3.18 Comparison results-1 on dynamic scenes from *PersonWalking* dataset [27]: (a) color frame, (b) raw depth frame, (c) result from Garcia et al. [34], (d) result from Avetisyan et al. [44], (e) our result. Our result perform well in removing the artifacts and preserving the sharpness of the edges both for static and dynamic parts of the frame. 54

3.19 Comparison results-2 on dynamic scenes from *PersonBox* dataset (1st and 2nd rows) and *PersonChair* dataset [47] (3rd row): (a,d,g) raw depth frames, (b,e,h) result from Islam et al. [47] (ghosting artifacts are visible in the rectangle-marked area and in other areas), (c,f,i) our results with gradient-based preprocessing (ghosting artifacts are removed). Our results in (c), (f) and (i) do not exhibit ghosting artifacts since we used the gradient-based spatial preprocessing. 55

3.20 Comparison results on two benchmark depth frames of *Book* and *Art* from Middlebury [87]: (a) depth frames with simulated degradation, (b) results from Camplani et al. [28], (c) results from Wang et al. [36] (d) our results. Our method yields sharper edges, while reducing the artifacts, than the other methods. 56

4.1 Images of our experimental setup for the steady structure: (a) steady structure with depth camera and test target – view-1, (b) steady structure with depth camera and test target – view-2, (c) 'L-shaped' ground plane, required by OptiTrack tracking system for getting the world coordinate system, (d) steady structure – the two-rig rail system on which the target moves to the marked positions. 63

4.2 Images of our experimental setup for the test target: (a) the test target attached on the moveable platform on the steady structure, (b) the test target mounted to the steady structure with the camera gear. 64

4.3 Images of our experimental setup for the depth camera: (a) the depth camera attached to the height-adjustable rig of the steady structure, (b) the depth camera mounted to the steady structure with the camera gear, (c,d) markers on the Kinect V1 and V2 depth cameras. 66

4.4 Illustration of our tracking setup along with placement of the 12 tracking cameras, recording depth camera, test target and the steady structure. . . . 66

4.5 Placement of markers on the depth camera and the test target and the position of part of the tracking cameras. Here, in (a) we can see that the test target (the checkerboard) is equipped with a asymmetric structure and another eight markers which provides a stable and steady position in the world coordinate system. Since the eight markers are on the same plane, we needed to attach the asymmetric structure to create a trackable rigid body for the OptiTrack system. A zoomed in version of the asymmetric structure in (b) shows both the front and behind view of the structure. The green marked marker is considered as the pivot of the rigid body. The depth cameras in (c) and (d) are equipped with several markers, along their length and width directions, which also helps to create a trackable rigid body for the OptiTrack system. Here the green marked markers, in (c) and (d) are placed on top of the estimated position of the depth sensor. We used this marker to create the pivot for this trackable rigid body. Finally, a part of the placement of the tracking cameras on the four-sided rig attached near the celling of the room are shown in (d). 67

4.6 Images from the Motive software of the OptiTrack tracking system showing the placement of the tracking cameras in (a) and the world coordinate system (the red, green and blue arrows represents X,Y and Z axes respectively) is shown inside the yellow marked circle in (a). We can see the rigid bodies (Kinect and CheckerBoard) in (b) which correspond to the depth camera and the test target respectively. In (b), we can see that the coordinate systems of both the rigid bodies align with the world coordinate system provided by the OptiTrack system. The yellow marked ellipse in (b) shows the location of the 'L-shaped' ground plane required by the OptiTRack system to define the world coordinate of the tracking system. 69

4.7 A side-by-side view of the ground truth image and recoded depth image. Here (a) shows the ground truth image drawn on an OpenGL window and (b) shows the recorded depth image on an OpenCV window. We can see both the images in (a) and (b) are similar in size, shape and orientation. . . 76

4.8 A side-by-side view of the binary images, using the depth thresholds, of the ground truth image and recoded depth image. Here (a) and (b) show the binary images of the ground truth image and the recorded depth image respectively. 76

4.9 A side-by-side view of the extracted regions from the ground truth image and recoded depth image. Here (a) and (b) show the extracted regions of the ground truth image and the recorded depth image respectively. 78

4.10 3D plot showing different views of the test target extracted from the recorded depth image. Here (a)–(f) show the front, back, top, bottom, left and right views of the extracted region of the recorded depth image. . . 78

4.11 3D plot showing different views of the test target extracted from the recorded depth image and the ground truth depth image. Here (a)–(f) show the front, back, top, bottom, left and right views of the extracted region of the recorded depth image and the ground truth depth image. Here, ground truth image is shown in gray color in (a)–(f). For example, in (c) the ground truth image is the gray surface plane which is passing through the recorded depth image. 79

4.12 Front and back view of the fitted planes through the extracted regions of the recorded depth image and the ground truth depth image; green and gray planes represent the recorded depth image and the ground truth depth respectively. Here (a) and (b) show the front and back views of the fitted planes of the extracted region of the recorded depth image and the ground truth image respectively. The zoomed image in the middle shows the very small distance between these planes. 80

4.13 Extracted noise mapped on the ground truth image at two different distances. Here (a) and (b) shows the noise mapped on the ground truth images created at distances 1.6 meters and 1.7 meters respectively. The color bar shows the scale of the amount of noise in millimetres (mm) over the surface of the ground truth images. We can see that the variation of the amount of noise in image (a) and (b) which are located at two different positions. 81

4.14 3D views of the recorded depth images and ground truth images of the test target at three different distances from the depth camera. Here, (a,e) show the 3D views of all the depth images and the ground truth images in one plot whereas, (b,c,d,f,g,h) show different plots showing different 3D views of the test target located at distances 1.4 meters, 1.6 meters and 1.8 meters from the depth camera respectively. We can see presence of a rather large hole in the middle of the depth images in (c) and (d) which were caused by reflections. As we move far away from the lighting source causing the reflection, the hole gets smaller and in (b) we do not perceive that hole anymore. We perceive a difference in the quantity of noise on the surface of the depth images as the target is moved farther from the camera by looking at the four holes (we placed four reflective objects on these positions so that we get some noise at these locations). By looking at images (d) to (b), we can see that the amount of noise varies (noise increases as the distance of the object from the camera increases) with varying distances of the test target from the camera. 83

4.15 3D views of the alignment of ground truth image and recorded depth images of the test target at three different distances (1.4 meters, 1.6 meters and 1.8 meters) from the depth camera. As we move from (d) to (b), we can perceive how the ground truth aligns and intersects with the image plane of the recorded depth images. 84

4.16 3D views of the noise obtained by subtracting the ground truth images from the recorded depth images of the test target at three different distances (1.4 meters, 1.6 meters and 1.8 meters) from the depth camera. Here, by looking from (d) to (b), we can see that the amount of noise changes as the target is moved far away from the depth camera. 85

4.17 3D views of the recorded depth image and ground truth image at three different viewing angles at one specific distance from the depth camera. Here, (a) shows the plots from all the three viewing angles and (b,c,d) show the plots for the rotation of the depth camera to 5° to the left side, 0° (no rotation i.e. perpendicular to the test target) and 4° to the right side respectively. We see here variations in depth values on the surface of the test target from one viewing angle to another. 86

4.18 3D views of the front and back side of the fitted planes through the extracted regions of the recorded depth image and the ground truth depth image at the three angles, mentioned in Figure 4.17, respectively; green and gray planes represent the recorded depth image and the ground truth depth image respectively. The zoomed images above (b,c,d) show how the fitted planes align with each other. Here, the zoomed parts of (a) and (b) show a small gap (less than 0.5 millimetres) between the fitted planes of the recorded depth image and the ground truth image, whereas (c) show no such distance. This gap occurs due to manually moving the test target from one location to another or manually rotating from one angle to another which seems to induce such gap. 87

4.19 3D views of the noise obtained by subtracting the ground truth images from the recorded depth images of the test target at three different viewing angles, mentioned in Figure 4.17, respectively. Here, by looking from (a) to (c), we can see that the amount of noise varies from one viewing angle to another, specifically at the bottom right corner of the test target shows the variations in the amount of noise in millimeter (mm). Moreover, the scales (shown at the right side of each images) in these three images have different maximum values which also indicate that the amount of noise varies from one angle to another. 87

4.20 Variations of the noise quantity at various distances of the object from the camera and at various viewing angle of the camera with respect to the target. Here, (a) and (b) show the amount of noise changed due to the change in distance of the target the camera and for the change in viewing angle of the camera. In (a), we can see that the lowest noise (marked with a green ellipse) occurs in between the range of 1 meter and 1.5 meters and it increases beyond this range. The error is lowest at this range because at this distance range, Kinects can estimate the depth values better. In (b), we can see that the lowest error occurs when the viewing angle between the camera and the target is 0 and then it becomes bigger with greater viewing angles. 88

4.21 Recorded depth image and ground truth image under two different lighting sources. Here, (a) to (c) show the color image, depth image and the 3D plot of the extracted region of the target and the ground truth image under room light (with color temperature of 5,500K). And, (d) to (f) show the color image, depth image and the 3D plot of the extracted region of the target and the ground truth image under studio light (with color temperature of 6,500K). Under these two lighting sources, although the color images show different shades of lighting, very little change is perceived in the depth images and on the 3D plots. 90

4.22 Absolute difference between the depth images under two lighting sources and between the noise on these two depth images. Here, (a) shows the absolute differences between the depth images under room light and studio light. By looking at the plot and the scale of variation on the right side, we perceive very small variation among these two images. And, (b)shows the absolute differences between the noise generated on the depth images under room light and studio light. Here too, we observe very little variation among the two noise quantities. 90

4.23 Bar diagrams showing the mean noise per pixel for the images recorded under room light and studio light. Here, (a) and (b) show the bar diagrams of the mean noise per pixel of the depth images under room light and studio light respectively. Although, we perceive little variations in the noise level for these two images, but the variations are not very drastic for the majority of the pixels. 91

4.24 Extracted test targets, located at the same position, from the depth images captured by Kinect V1 and Kinect V2 sensors. Here, (a) and (b) show the test targets by Kinect V1 and V2 respectively. We can see the squares in (b) (also in Figure 4.11(a) and (b)) which correspond to the actual black and white squares on the test target (checkerboard) we used for the experiments. Although these squares are on the same plane, Kinect V2 is showing different depth values for these squares. We don't see such squares on the depth image of Kinect V1 in (a). 91
4.25 Top and bottom views of the extracted test target in 3D showing the depth values. Here, (a) and (b) show the 3D views of the top side of the test target extracted from the depth images captured by Kinect V1 and V2 respectively. We can see that the depth values are fluctuating within a relatively short range for Kinect V1 than for Kinect V2. For Kinect V2, we see quite large fluctuation of depth value compared to the fluctuation in Kinect V1. We also perceive similar behaviour of the depth values for the bottom view of the extracted test target in (c) and (d). Here, we can see even more spikes indicating more fluctuations in the depth values for Kinect V2 than Kinect V1. 92

5.1 A communication system with multi-camera setup. 96
5.2 Color degradation strategy for the frames. 100
5.3 Amount of color reduction:(a) Mother and daughter MD (b) Coastguard CG. Both the *a* versus b** scatterplots and the CIELAB diagrams for the NKFs of (a) and (b) show that the degraded NKFs contain less color information than the original NKFs. 101
5.4 Diagram of proposed enhancement methods. 104
5.5 Enhancement results for Coastguard CG sequence: (a) Proposed method 1, (b) Proposed method 2. 105
5.6 Enhancement results with proposed method 1 on: (a) Mother and daughter MD sequence, (b) *ice* sequence. 106
5.7 Enhancement results varying KF frequency for Coastguard CG sequence: (a) Proposed method 1, (b) Proposed method 2. 106
5.8 Enhancement results varying KF frequency with proposed method 1 on: (a) MD sequence, (b) *ice* sequence. 107
5.9 PSNR result for MD sequence: varying total no. of frames. 107
5.10 Subjective result comparison for frame 12 of MD and CG datasets: (a,e) decoded only frame of MD and CG, (b,f) *inverse process* of degradation (Lab enhanced; see last paragraph in Section 5.3.1.1) frame of MD and CG, (c,g) enhanced, by our methods, frame of MD and CG, (d,h) non-degraded reference coder frame of MD and CG. 108

5.11 Light field capture, processing and displaying pipeline. 110

5.12 Sample scene acquisition and reconstruction for light field display: (a) sample light field capturing, (b) light field reconstruction. 111

5.13 Proposed data reduction strategy: (a) calculated significant camera image pixels from a sample capture, (b) percentage of pixels from each camera image used in sample light field reconstruction. 112

6.1 **Illustration of using our gSMOOTH in a telepresence scenario.** The depth image on the right screen of the local site is enhanced by gSMOOTH while the left depth image shows artifacts (since the processing of the frames inside the sliding window is in progress). 118

6.2 **Illustration of sliding window of frames in gSMOOTH.** Here, there are three frames inside the sliding window, hence we will perceive the enhanced output from the fourth frame. 118

6.3 **Effect of applying our depth enhancement approach on 3D reconstruction of two scenes**. The images on the top row are from the first scene where a person is standing and the images on the bottom row are from the second scene where a person is sitting on a desk. Here, the 3D reconstruction in (a) is using the raw depth frames to reconstruct the scene, while the 3D reconstructions in (b) and (c) use a median filter and our proposed depth enhancement approach respectively. The images in (c) show clear improvement on different parts of the scene, such as the body of the person and the floor areas, in comparison with the images in (b) where a simple median filter is used to enhance the depth frames. 120

6.4 **Our depth enhancement result on 3D reconstruction**: the raw depth frames on top images shows holes while our outputs removes the hole significantly. The enhancement is reflected on the 3D reconstructed images where the 3D reconstruction on the right side shows improved surface than the 3D reconstruction on the left side. The respective zoomed parts show the detailed enhancement on the surface of the 3D reconstruction. 121

6.5 **Effect of applying our depth enhancement approach on 3D reconstruction of a person's upper body part**. Here, the surface reconstruction in (b) using the enhanced depth frames by our approach shows significant improvement in comparison with the surface reconstruction in (a) which uses the raw depth frames. 122

6.6 **Illustration of recording a lecture for an e-learning system.** The lecture is being recorded by camera and the content is being displayed on screen using a projector. 124

6.7 **Hand gesture detection and enhancement within a lecture recording session in an e-learning system.** Here, a random frame in (a) shows that there are quite a lot of holes appearing in different parts of the captured scene. The detected hand in the bottom image of (a) also shows a lot of holes on it which potentially would affect the performance of gesture recognition. Enhancement methods, such as morphological closing (b) and interpolation technique (c) remove these artifacts partially, however, these outputs do not preserve the homogeneity of the edges of the fingers. Our method in (d) is able to remove a major part of the holes. 125

List of Tables

2.1 A review of advantages and limitation of different depth image enhancement strategies. 22
2.2 Capability of the temporal and spatio-temporal methods regarding addressing the research questions yet to be solved. 32

3.1 Performance of gSMOOTH in frames per second (*fps*) on the five test depth sequences. 52
3.2 PSNR and SSIM scores on the results of methods [28,36] and our gSMOOTH on benchmark depth frames. 57

4.1 Change in the angle between the fitted planes through the recorded depth image and the ground truth image for three viewing angles. 85

Appendices

Appendix A

Matlab codes for generating ground plane and visualising in 3D

A.1 Codes to create binary images, extract test targets and plot them in 3D

```
clear;
capturedImage = double(imread('CapturedImage.png'));
groundTruthImage = double(imread('GroundTruthImage.png'));
surf(capturedImage,'EdgeColor','none');
Up_Limit=1460;
Low_Limit= 1300;
binaryImagefromCapturedImage = capturedImage > Low_Limit & ↘
    capturedImage < Up_Limit ;
binaryImageFromGrounTruthImage= groundTruthImage > 0; ;

propsCapturedImage = regionprops(binaryImagefromCapturedImage, ↘
    'BoundingBox');
boundingBoxCapturedImage = propsCapturedImage.BoundingBox

testTarget_CapturedImage = imcrop(CapturedImage, ↘
    boundingBoxCapturedImage);

propsGrounTruthImage = regionprops(binaryImageFromGrounTruthImage, ↘
    'BoundingBox');
boundingBoxGrounTruthImage = propsGrounTruthImage.BoundingBox

testTarget_GrounTruthImage = imcrop(groundTruthImage, ↘
    boundingBoxGrounTruthImage);

figure
surf(testTarget_GrounTruthImage, 'EdgeColor', [0.5 0.5 0.5]);
```

```
hold on
surf(testTarget_CapturedImage, 'EdgeColor', 'none');
hold off
```

Listing A.1 – Matlab codes for creating binary images, extracting test targets and plotting them in 3D

A.2 Matlab codes for fitting planes to the captured depth image

```

% for captured image

x = 1:size(testTarget_CapturedImage,1);
y = 1:size(testTarget_CapturedImage,2);
z= testTarget_CapturedImage;
[xo,yo,zo] = prepareSurfaceData(x,y,z);


fc1 =fit([yo,xo],zo,'poly10','Normalize','on','Robust','Bisquare');


%for ground truth image
x1 = 1:size(testTarget_GrounTruthImage,1);
y1 = 1:size(testTarget_GrounTruthImage,2);
z1= testTarget_GrounTruthImage;
[xg,yg,zg] = prepareSurfaceData(x1,y1,z1);

fg = fit([yg,xg],zg,'poly10','Normalize','on','Robust','Bisquare');


figure
h1=plot(fc1);
set(h1(1),'Edgecolor','none')
set(h1(1),'FaceAlpha','0.99')
set(h1(1),'FaceColor','green')

hold on
h2=plot (fg);
set(h2(1),'Edgecolor','none')
set(h2(1),'FaceAlpha','0.99')
set(h2(1),'FaceColor',[0.5 0.5 0.5])

view(5, -19);
hold off
```

Listing A.2 – Matlab codes for fitting plane through the captured and ground truth depth images

A.3 Matlab codes for extracting noise and mapping it onto the ground truth image

```

noise = ↘
    imabsdiff(testTarget_CapturedImage,testTarget_GrounTruthImage);


%mapping the noise onto the ground truth image
figure

surf(testTarget_GrounTruthImage, noise, 'FaceColor', 'texturemap', ↘
    'EdgeColor', 'none');
colormap(jet);
view(2, 46);
colorbar
```

Listing A.3 – Matlab codes for extracting noise and mapping it onto the ground truth image

Publications

Scientific Journals (peer reviewed):

- ABM Tariqul Islam, Christian Scheel, Renato Pajarola and Oliver Staadt, Robust Enhancement of Depth Images from Depth Sensors, Computers & Graphics Journal, Volume 68, Pages 53-65, 2017.

- ABM Tariqul Islam, Jonas Flint, Philipp Jaecks and Clemens H Cap, A proficient and versatile online student-teacher collaboration platform for large classroom lectures, International Journal of Educational Technology in Higher Education, 2017.

- ABM Tariqul Islam, Ju Bin Song, "Enhancement of the Detection Probability for Distributed Cooperative Spectrum Sensing using UWB as a Common Channel", IEEK (Institute of Electronics Engineers of Korea), South Korea, pp: 22-31, July 2008.

- Nilimesh Halder, ABM Tariqul Islam and Ju Bin Song, "Modeling and Formal Verification of Communication Protocols for Remote Procedure Call", International Journal of Computer Science and Network Security (IJCSNS) vol.7 No. 7, pp: 63- 71, July 2007.

Scientific Conferences (peer reviewed):

- ABM Tariqul Islam, Martin Luboschik, Anton Jirka and Oliver Staadt. gSMOOTH - A Gradient based Spatial and Temporal Method of Depth Image Enhancement. InProceedings Computer Graphics International (CGI)'18, Bintan, Indonesia, 2018.

- ABM Tariqul Islam, Jonas Flint, Philipp Jaecks, Clemens H Cap: Multiscript - an online student-teacher collaboration platform for classroom lectures. Lecture Notes in Informatics (LNI), Gesellschaft fur Informatik, Bonn, 2016.

- Christian Scheel, ABM Tariqul Islam, Oliver Staadt: An Efficient Interpolation Approach for Low Cost Unrestrained Gaze Tracking in 3D Space, ICAT-EGVE - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments, 2016.

- ABM Tariqul Islam, Renato Pajarola, Christian Scheel and Oliver Staadt. Depth Image Enhancement using 1D Least Median of Squares. InProceedingsComputer Graphics International (CGI)'15, Strasbourg, France, 2015.
- ABM Tariqul Islam, Christian Scheel, Renato Pajarola and Oliver Staadt. Robust Enhancement of Depth Images from Kinect Sensor. In Proceedings IEEE Virtual Reality Conference, Arles, France, 2015.
- ABM Tariqul Islam, Christian Scheel, Ali Shariq Imran and Oliver Staadt. Fast and Accurate 3D Reproduction of a Remote Collaboration Environment. InProceedings Virtual, Augmented and Mixed Reality, Lecture Notes in Computer Science, Springer International Publishing, 8525, pp.351-362, June 2014.
- ABM Tariqul Islam and Oliver Staadt. Bandwidth-Efficient Image Degradation and Enhancement Model for Multi-Camera Telepresence Environments. In: European Conference on Visual Media Production (CVMP). London, UK, November 2013.
- Adhikarla, V.K.; Tariqul Islam, ABM.; Kovacs, P.T.; Staadt, O., "Fast and efficient data reduction approach for multi-camera light field display telepresence systems," 3DTV-Conference: The True Vision-Capture, Transmission and Dispaly of 3D Video (3DTV-CON), 2013 , vol., no., pp.1,4, 7-8 Oct. 2013
- ABM Tariqul Islam, Stephan Ohl, and Oliver Staadt, "Multi-Camera Acquisition and Placement Strategy for Displaying High-Resolution Images for Telepresence Systems", in proceedings of Eurographics Poster papers, pp. 13-14. 2013.
- ABM Tariqul Islam, and Ivar Farup, "Spatio-temporal colour correction of strongly degraded movies" Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7866, 78660Z, 2011, USA.
- ABM Tariqul Islam and Ivar Farup, "Enhancing the output of spatial color algorithms," 2nd European workshop on visual information processing, EUVIP, Paris, France, 7-12 (2010).
- Razibul Islam, A.H.M. Town, G. Tariqul Islam, A.B.M, "PAPR and SFDR of an OFDM-RoF link", Opto-Electronics and Communications Conference, 2008 and the 2008 Australian Conference on Optical Fibre Technology. OECC/ACOFT 2008. Joint conference. Publication Date: 7-10 July 2008, Australia.
- ABM Tariqul Islam, Md. Imrul Hassan, AHM Razibul Islam and Ju Bin Song, "Location Estimation in Distributed Ad-Hoc Wireless Sensor Networks", The 22nd International Technical Conference on Circuits/Systems, Computer and Communications (ITC-CSCC) pages: 701-702 , July-2007, Busan, South Korea.

Bibliography

[1] F. Garcia Becerro, “Sensor fusion combining 3-d and 2-d for depth data enhancement,” Ph.D. dissertation, University of Luxembourg, Luxembourg, 2012. [Online]. Available: http://hdl.handle.net/10993/10633

[2] A. Atapour-Abarghouei and T. P. Breckon, “A comparative review of plausible hole filling strategies in the context of scene depth image completion,” *Computers & Graphics*, vol. 72, pp. 39 – 58, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849318300219

[3] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, Jul 1990.

[4] D. Miao, J. Fu, Y. Lu, S. Li, and C. W. Chen, “Texture-assisted kinect depth inpainting,” in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 604–607.

[5] K. R. Vijayanagar, M. Loghman, and J. Kim, “Real-time refinement of kinect depth maps using multi-resolution anisotropic diffusion,” *Mobile Networks and Applications*, vol. 19, no. 3, pp. 414–425, Jun 2014. [Online]. Available: https://doi.org/10.1007/s11036-013-0458-7

[6] C. Chen, J. Cai, J. Zheng, T. J. Cham, and G. Shi, “Kinect depth recovery using a color-guided, region-adaptive, and depth-selective framework,” *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 2, pp. 12:1–12:19, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2700475

[7] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, “Color-guided depth recovery from rgb-d data using an adaptive autoregressive model,” *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3443–3458, Aug 2014.

[8] L. Sheng and K. N. Ngan, “Depth enhancement based on hybrid geometric hole filling strategy,” in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 2173–2176.

[9] S. Liu, Y. Wang, J. Wang, H. Wang, J. Zhang, and C. Pan, “Kinect depth restoration via energy minimization with tv21 regularization,” in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 724–724.

[10] D. Herrera C., J. Kannala, L. Ladický, and J. Heikkilä, “Depth map inpainting under a second-order smoothness prior,” in *Image Analysis*, J.-K. Kämäräinen and M. Koskela, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 555–566.

[11] S. Lee, “Depth camera image processing and applications,” in *2012 19th IEEE International Conference on Image Processing*, Sept 2012, pp. 545–548.

[12] R. Lange and P. Seitz, “Solid-state time-of-flight range camera,” *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, Mar 2001.

[13] S. J. Miller, “The method of least squares,” *Mathematics Department, Brown University*, pp. 1 – 7, 2006.

[14] H. Abdi, “The method of least squares,” *In Neil Salkind (Ed.), Encyclopedia of Research Design.*, pp. 1–7, 2007.

[15] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[16] A. Telea, “An image inpainting technique based on the fast marching method,” *Graphics Tools*, vol. 9, no. 1, pp. 25–36, 2004.

[17] A. Atapour-Abarghouei, G. P. de La Garanderie, and T. P. Breckon, “Back to butterworth - a fourier basis for 3d surface relief hole filling within rgb-d imagery,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 2813–2818.

[18] S. H. Baek, I. Choi, and M. H. Kim, “Multiview image completion with space structure propagation,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 488–496.

[19] L. Wang, S. Piao, Y. Jiang, and L. Zhang, “On a web-graph-based micronetwork architecture for socs,” *International Journal of Computers and Applications*, vol. 30, no. 1, pp. 1–8, 2008. [Online]. Available: https://doi.org/10.1080/1206212X.2008.11441879

[20] A. Hervieu, N. Papadakis, A. Bugeau, P. Gargallo, and V. Caselles, “Stereoscopic image inpainting: Distinct depth maps and images inpainting,” in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 4101–4104.

[21] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, “Image Melding: Combining inconsistent images using patch-based synthesis,” *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012)*, vol. 31, no. 4, pp. 82:1–82:10, 2012.

[22] A. Efros and T. Leung, “Texture synthesis by non-parametric sampling,” *IEEE International Conference on Computer Vision*, pp. 1033–1038, 1999, cited By 5.

[23] A. Criminisi, P. Perez, and K. Toyama, “Region filling and object removal by exemplar-based image inpainting,” *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, Sept 2004.

[24] S. Lu, X. Ren, and F. Liu, “Depth enhancement via low-rank matrix completion,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 3390–3397.

[25] L. Chen, H. Lin, and S. Li, “Depth image enhancement for kinect using region growing and bilateral filter,” in *21st International Conference on Pattern Recognition*, Nov 2012, pp. 3070–3073.

[26] Q. Yang, N. Ahuja, R. Yang, K.-H. Tan, J. Davis, B. Culbertson, J. Apostolopoulos, and G. Wang, “Fusion of median and bilateral filtering for range image upsampling,” *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4841–4852, Dec 2013.

[27] M. Camplani, T. Mantecon, and L. Salgado, “Depth-color fusion strategy for 3-d scene modeling with kinect,” *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1560–1571, 2013.

[28] M. Camplani and L. Salgado, “Efficient spatio-temporal hole filling strategy for kinect depth maps,” in *Proc. SPIE*, vol. 8290, 2012, pp. 82 900E–82 900E–10.

[29] J. Shen and S. C. S. Cheung, “Layer depth denoising and completion for structured-light rgb-d cameras,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 1187–1194.

[30] K. He, J. Sun, and X. Tang, “Guided image filtering,” in *11th European Conference on Computer Vision (ECCV), Springer Berlin Heidelberg*, 2010, pp. 1–14.

[31] N. E. Yang, Y. G. Kim, and R. H. Park, “Depth hole filling using the depth distribution of neighboring regions of depth holes in the kinect sensor,” in *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*, Aug 2012, pp. 658–661.

[32] A. Maimone and H. Fuchs, “Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras,” in *10th IEEE ISMAR*, October 2011, pp. 137–146.

[33] X. Xu, L.-M. Po, K.-H. Ng, L. Feng, K.-W. Cheung, C.-H. Cheung, and C.-W. Ting, “Depth map misalignment correction and dilation for DIBR view synthesis,” *Signal Processing: Image Communication*, vol. 28, no. 9, pp. 1023 – 1045, 2013.

[34] F. Garcia, D. Aouada, T. Solignac, B. Mirbach, and B. Ottersten, “Real-time depth enhancement by fusion for rgb-d cameras,” *Computer Vision, IET*, vol. 7, no. 5, pp. 1–11, October 2013.

[35] C. Chen, J. Cai, J. Zheng, T.-J. Cham, and G. Shi, “A color-guided, region-adaptive and depth-selective unified framework for kinect depth recovery,” in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, Sept 2013, pp. 007–012.

[36] Z. Wang, J. Hu, S. Wang, and T. Lu, “Trilateral constrained sparse representation for kinect depth hole filling,” *Pattern Recognition Letter*, vol. 65, no. C, pp. 95–102, Nov. 2015.

[37] J. Hu, H. Tang, and K. C. Tan, “A hierarchical organized memory model using spiking neurons,” in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug 2013, pp. 1–6.

[38] Y. J., Y. X., L. K., and H. C., “Depth recovery using an adaptive color-guided auto-regressive model,” in *European Conference on Computer Vision (ECCV)*, 2012, pp. 158–171.

[39] F. Qi, J. Han, P. Wang, G. Shi, and F. Li, “Structure guided fusion for depth map inpainting,” *Pattern Recognition Letter*, vol. 34, no. 1, pp. 70–76, Jan. 2013.

[40] J. Liu, X. Gong, and J. Liu, “Guided inpainting and filtering for kinect depth maps,” in *21st International Conference on Pattern Recognition (ICPR)*, Nov 2012, pp. 2055–2058.

[41] D. Miao, J. Fu, Y. Lu, S. Li, and C. W. Chen, “Texture-assisted kinect depth inpainting,” in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 604–607.

[42] K. R. Vijayanagar, M. Loghman, and J. Kim, “Real-time refinement of kinect depth maps using multi-resolution anisotropic diffusion,” *Mobile Networks and Applications*, vol. 19, no. 3, pp. 414–425, Jun 2014. [Online]. Available: https://doi.org/10.1007/s11036-013-0458-7

[43] X. Xu, L.-M. Po, C.-H. Cheung, L. Feng, K.-H. Ng, and K.-W. Cheung, “Depth-aided exemplar-based hole filling for DIBR view synthesis,” in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2013, pp. 2840–2843.

[44] R. Avetisyan, C. Rosenke, M. Luboschik, and O. Staadt, "Temporal filtering of depth images using optical flow," in *Proceedings of the 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'16)*, 2016.

[45] T.-W. Hui and K. N. Ngan, "Motion-depth: Rgb-d depth map enhancement with motion and depth in complement," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014, pp. 3962–3969.

[46] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by kinect depth camera," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2011, pp. 1–4.

[47] A. T. Islam, C. Scheel, R. Pajarola, and O. Staadt, "Robust enhancement of depth images from depth sensors," *Computers & Graphics*, vol. 68, no. Supplement C, pp. 53 – 65, 2017.

[48] D. Fu, Y. Zhao, and L. Yu, "Temporal consistency enhancement on depth sequences," in *28th Picture Coding Symposium*, Dec 2010, pp. 342–345.

[49] Y. Berdnikov and D. Vatolin, "Real-time depth map occlusion filling and scene background restoration for projected-pattern-based depth cameras," in *Proceedings of the 21st international conference on computer graphics and vision (GraphiCon'11)*, 2011.

[50] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "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*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 559–568. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047270

[51] S.-Y. Kim, J.-H. Cho, A. Koschan, and M. Abidi, "Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor," in *20th International Conference on Pattern Recognition (ICPR)*, Aug 2010, pp. 2358–2361.

[52] K. Xu, J. Zhou, and Z. Wang, "A method of hole-filling for the depth map generated by kinect with moving objects detection," in *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting*, June 2012, pp. 1–5.

[53] M. Camplani and L. Salgado, "Adaptive spatio-temporal filter for low-cost camera depth maps," in *2012 IEEE International Conference on Emerging Signal Processing Applications*, Jan 2012, pp. 33–36.

[54] J. Wang, P. An, Y. Zuo, Z. You, and Z. Zhang, "High accuracy hole filling for kinect depth maps," in *Proc.SPIE*, vol. 9273, 2014, pp. 9273 – 9273 – 17. [Online]. Available: https://doi.org/10.1117/12.2071437

[55] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, and C. Theobalt, "Coherent spatiotemporal filtering, upsampling and rendering of rgbz videos," *Computer Graphics Forum*, vol. 31, no. 2, pp. 247–256, 2012.

[56] A. Amamra and N. Aouf, "Gpu-based real-time rgbd data filtering," *Journal of Real-Time Image Processing*, pp. 1–18, 2014.

[57] S. Bhattacharya, S. Gupta, and K. Venkatesh, "High accuracy depth filtering for kinect using edge guided inpainting," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2014, pp. 868–874.

[58] L. Zhang, P. Shen, S. Zhang, J. Song, and G. Zhu, "Depth enhancement with improved exemplar-based inpainting and joint trilateral guided filtering," in *IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 4102–4106.

[59] S. Beck, A. Kunert, A. Kulik, and B. Froehlich, "Immersive group-to-group telepresence," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 4, pp. 616–625, April 2013.

[60] J. Dolson, J. Baek, C. Plagemann, and S. Thrun, "Upsampling range data in dynamic environments," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010.

[61] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 664–672, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015777

[62] S. Liu, P. Lai, D. Tian, C. Gomila, and C. W. Chen, "Joint trilateral filtering for depth map compression," in *Proc.SPIE*, vol. 7744, 2010, pp. 7744 – 7744 – 10. [Online]. Available: https://doi.org/10.1117/12.863341

[63] J. Wasza, S. Bauer, and J. Hornegger, "Real-time preprocessing for dense 3-d range imaging on the gpu: Defect interpolation, bilateral temporal averaging and guided filtering," in *International Conference on Computer Vision Workshops*, Nov 2011, pp. 1221–1227.

[64] Q. H. Nguyen, M. N. Do, and S. J. Patel, "Depth image-based rendering from multiple cameras with 3d propagation algorithm," in *Proceedings of the 2Nd International Conference on Immersive Telecommunications*, ser. IMMERSCOM '09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences,

Social-Informatics and Telecommunications Engineering), 2009, pp. 6:1–6:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1594108.1594116

[65] D. Min, J. Lu, and M. Do, "Depth video enhancement based on weighted mode filtering," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1176–1190, 2012, cited By 155.

[66] I. Daribo, C. Tillier, and B. Pesquet-Popescu, "Distance dependent depth filtering in 3d warping for 3dtv," in *2007 IEEE 9th Workshop on Multimedia Signal Processing*, Oct 2007, pp. 312–315.

[67] W.-Y. Chen, Y.-L. Chang, S.-F. Lin, L.-F. Ding, and L.-G. Chen, "Efficient depth image based rendering with edge dependent depth filter and interpolation," in *IEEE International Conference on Multimedia and Expo, ICME 2005*, 2005, pp. 1314–1317.

[68] P. Lai, D. Tian, and P. Lopez, "Depth map processing with iterative joint multilateral filtering," in *28th Picture Coding Symposium*, Dec 2010, pp. 9–12.

[69] M. Mueller, F. Zilly, and P. Kauff, "Adaptive cross-trilateral depth map filtering," in *2010 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*, June 2010, pp. 1–4.

[70] V. Garro, C. d. Mutto, P. Zanuttigh, and G. M. Cortelazzo, "A novel interpolation scheme for range data with side information," in *2009 Conference for Visual Media Production*, Nov 2009, pp. 52–60.

[71] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sep 2004.

[72] F. Meyer, "Color image segmentation," in *1992 International Conference on Image Processing and its Applications*, 1992, pp. 303–306.

[73] A. Atapour-Abarghouei and T. Breckon, "Depthcomp : real-time depth image completion based on prior semantic scene segmentation." in *28th British Machine Vision Conference (BMVC) 2017*. British Machine Vision Association (BMVA), September 2017.

[74] L. M. Po, S. Zhang, X. Xu, and Y. Zhu, "A new multidirectional extrapolation hole-filling method for depth-image-based rendering," in *2011 18th IEEE International Conference on Image Processing*, Sept 2011, pp. 2589–2592.

[75] T. Matsuo, N. Fukushima, and Y. Ishibashi, "Weighted joint bilateral filter with slope depth compensation filter for depth map refinement," in *VISAPP 2013 - Proceedings*

of the International Conference on Computer Vision Theory and Applications, vol. 2, 2013, pp. 300–309.

[76] S. W. Jung, “Enhancement of image and depth map using adaptive joint trilateral filter,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 258–269, Feb 2013.

[77] M. M. O. B. B. Richard and M. Y.-S. Chang, “Fast digital image inpainting,” in *International Conference on visualization, imaging and image processing*, 2001, pp. 106–111.

[78] A. T. Islam, C. Scheel, R. Pajarola, and O. Staadt, “Depth image enhancement using 1d least median of squares,” in *Computer Graphics International (CGI’15)*, June 2015.

[79] ——, “Robust enhancement of depth images from kinect sensor,” in *2015 IEEE Virtual Reality (VR)*, March 2015, pp. 197–198.

[80] L. Sheng, K. N. Ngan, and S. Li, “Temporal depth video enhancement based on intrinsic static structure,” in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 2893–2897.

[81] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral upsampling,” *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276497

[82] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, “The lumigraph,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’96. New York, NY, USA: ACM, 1996, pp. 43–54. [Online]. Available: http://doi.acm.org/10.1145/237170.237200

[83] A. T. Islam, M. Luboschik, A. Jirka, and O. Staadt, “gsmooth: A gradient based spatial and temporal method of depth image enhancement,” in *Proceedings of Computer Graphics International 2018*, ser. CGI 2018. New York, NY, USA: ACM, 2018, pp. 175–184. [Online]. Available: http://doi.acm.org/10.1145/3208159.3208166

[84] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987.

[85] K. Khoshelham and E. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[86] C. Nguyen, S. Izadi, and D. Lovell, “Modeling kinect sensor noise for improved 3d reconstruction and tracking,” in *2nd International Conference on 3D Imaging,*

Modeling, Processing, Visualization and Transmission (3DIMPVT), Oct 2012, pp. 524–530.

[87] MiddleburyDatasets, http://vision.middlebury.edu/stereo/data/, 2013, accessed: 16-01-2018.

[88] M. K. Park, Y. Y. Lee, I. Y. Jang, and K. H. Lee, "Enhancement of range data using a structure-aware filter," *Computer Graphics*, vol. 48, no. C, pp. 48–59, May 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2015.02.006

[89] W. Zhou, A. Lumsdaine, L. Lin, W. Zhang, and R. Wang, "Edge-aware light-field flow for depth estimation and occlusion detection," *Electronic Imaging*, vol. 2017, no. 17, pp. 94–99, 2017.

[90] S. Kahn, H. Wuest, and D. Fellner, "Time-of-flight based scene reconstruction with a mesh processing tool for model based camera tracking." in *VISAPP 2010 - Proceedings of the International Conference on Computer Vision Theory and Applications*, vol. 1, 01 2010, pp. 302–309.

[91] I. atutomotive systems., http://www.ibeo-as.com, 2018, accessed: 01-07-2018.

[92] L. The Nipon Signal Co., http://www.signal.co.jp, 2018, accessed: 01-07-2018.

[93] O. tracking system, http://optitrack.com/, 2018, accessed: 26-09-2018.

[94] V. R. P. Network(VRPN), https://github.com/vrpn/vrpn, 2018, accessed: 27-09-2018.

[95] O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Computer Vision – ACCV 2016 Workshops*, C.-S. Chen, J. Lu, and K.-K. Ma, Eds. Cham: Springer International Publishing, 2017, pp. 34–45.

[96] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the Art on 3D Reconstruction with RGB-D Cameras," *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*, vol. 37, no. 2, 2018.

[97] V. K. Adhikarla, A. T. Islam, P. T. Kovacs, and O. Staadt, "Fast and efficient data reduction approach for multi-camera light field display telepresence systems," in *2013 3DTV Vision Beyond Depth (3DTV-CON)*, Oct 2013, pp. 1–4.

[98] T. Islam and O. Staadt, "Bandwidth-efficient image degradation and enhancement model for multi-camera telepresence environments," in *Proceedings of the 10th European Conference on Visual Media Production*, ser. CVMP '13. New York, NY, USA: ACM, 2013, pp. 14:1–14:8. [Online]. Available: http://doi.acm.org/10.1145/2534008.2534015

[99] B. Petit, J.-D. Lesage, C. Menier, J. Allard, J.-S. Franco, B. Raffin, E. Boyer, and F. Faure, “Multicamera real-time 3d modeling for telepresence and remote collaboration,” *International Journal of Digital Multimedia Broadcasting*, vol. 2010, pp. 247 108–12, 2009.

[100] M. Willert, S. Ohl, and O. G. Staadt, “Reducing bandwidth consumption in parallel networked telepresence environments,” in *VRCAI*, December 2012, pp. 247–254.

[101] E. Lamboray, S. Wurmlin, and M. Gross, “Data streaming in telepresence environments,” *IEEE Tran. on Visualization and Comp. Graphics*, vol. 11, no. 6, pp. 637–648, Jan. 2005.

[102] A. Maimone and H. Fuchs, “A first look at a telepresence system with room-sized real-time 3d capture and life-sized tracked display wall,” in *ICAT*, November 2011.

[103] J.-M. Lien, G. Kurillo, and R. Bajcsy, “Multi-camera tele-immersion system with real-time model driven data compression,” *The Visual Computer*, vol. 26, no. 1, pp. 3–15, Jan. 2010.

[104] F. Brandi, R. de Queiroz, and D. Mukherjee, “Super-resolution of video using key frames and motion estimation,” in *IEEE ICIP*, October 2008, pp. 321–324.

[105] M. Shen, P. Xue, and C. Wang, “Down-sampling based video coding using super-resolution technique,” *IEEE Tran. on Circuits and Systems for Video Technology*, vol. 21, no. 6, pp. 755–765, June 2011.

[106] Z. Hu, H. Li, and W. Li, “An adaptive down-sampling based video coding with hybrid super-resolution method,” in *IEEE ISCAS*, May 2012, pp. 504–507.

[107] E. H. Land and J. J. McCann, “Lightness and retinex theory,” *Journal of Optical Society of America*, vol. 61, no. 1, pp. 1–11, 1971.

[108] A. Rizzi, C. Gatta, and M. Daniele, “A new algorithm for unsupervised global and local color correction,” *Pattern Recognition Letters*, vol. 24, no. 11, pp. 1663–1677, July 2003.

[109] A. Jones, I. McDowall, H. Yamada, M. Bolas, and P. Debevec, “Rendering for an interactive 360° light field display,” *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276427

[110] M. Magnor and B. Girod, “Data compression for light-field rendering,” *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 10, no. 3, pp. 338–343, Apr. 2000. [Online]. Available: http://dx.doi.org/10.1109/76.836278

[111] V. Cheung, *Uniform Color Spaces*, J. Chen, W. Cranton, and M. Fihn, Eds. Cham: Springer International Publishing, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-14346-0_14

[112] TestDataSets, https://media.xiph.org/video/derf/, 2013, accessed: 14-04-2013.

[113] A. T. Islam, C. Scheel, A. S. Imran, and O. Staadt, "Fast and accurate 3d reproduction of a remote collaboration environment," in *Virtual, Augmented and Mixed Reality. Designing and Developing Virtual and Augmented Environments*, R. Shumaker and S. Lackey, Eds. Cham: Springer International Publishing, 2014, pp. 351–362.

[114] A. T. Islam, J. Flint, P. Jaecks, and C. H. Cap, "A proficient and versatile online student-teacher collaboration platform for large classroom lectures," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 1, p. 29, Nov 2017. [Online]. Available: https://doi.org/10.1186/s41239-017-0067-9

[115] M. Dou and H. Fuchs, "Temporally enhanced 3d capture of room-sized dynamic scenes with commodity depth cameras," in *IEEE Virtual Reality (VR)*, March 2014, pp. 39–44.

[116] M. Winkler, K. M. Höver, and M. Mühlhäuser, "A depth camera based approach for automatic control of video cameras in lecture halls," *Interactive Technology and Smart Education*, vol. 11, no. 3, pp. 169–183, 2014. [Online]. Available: https://doi.org/10.1108/ITSE-06-2014-0012

[117] B. Engelbert, C. Greweling, and K. Morisse, "The use and benefit of a xbox kinect based tracking system in a lecture recording service," in *Proceedings of EdMedia + Innovate Learning 2013*, J. Herrington, A. Couros, and V. Irvine, Eds. Victoria, Canada: Association for the Advancement of Computing in Education (AACE), June 2013, pp. 179–184. [Online]. Available: https://www.learntechlib.org/p/111952

[118] E. Lopes, J. Caetano, A. Abreu, and F. Grilo, "ireclass-an automatic system for recording classes," *arXiv preprint arXiv:1501.00149*, 2014.

[119] M. Asad and C. Abhayaratne, "Kinect depth stream pre-processing for hand gesture recognition," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 3735–3739.

[120] D. J. Ryan, "Finger and gesture recognition with microsoft kinect," Master's thesis, University of Stavanger, Norway, 2012.

[121] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," *arXiv preprint arXiv:1512.04134*, 2015.