UNIVERSITY OF CALIFORNIA, SAN DIEGO

Accurate, Efficient, and Robust 3D Reconstruction of Static and Dynamic Objects

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Kyoung-Rok Lee

Committee in charge:

Professor Truong Q. Nguyen, Chair
Professor Serge J. Belongie
Professor Pamela C. Cosman
Professor William S. Hodgkiss
Professor Jurgen P. Schulze

2014

The dissertation of Kyoung-Rok Lee is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

____

____

____

____

____
Chair

University of California, San Diego

2014

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to give my special thanks to my advisor Professor Truong Nguyen for his guidance and support. His guidance and encouragement allow me to challenge myself in research, and his insightful supports inspire me in every academic aspect.

I would like to also give my gratitude to other committee members Professors Serge Belongie, Pamela Cosman, William Hodgkiss, and Jurgen Schulze for their time and support in my research. I would like to thank colleagues in the video processing lab: Haleh Azartash, Can Bal, Stanley Chan, Kris Gibson, Yunghuan Hsieh, Natan Jacobson, Ankit Jain, Jason Juang, Byeong-Keun Kang, Ramsin Khoshabeh, Yeejin Lee, Zuchel Lee, Lester Liu, Enming Luo, Subarna Tripathi, Yujia Wang, and Menglin Zeng. It has been a pleasure and an honor to be a member and a colleague of this lab.

I would like to dedicate this dissertation and my accomplishment to my loving grandparents and parents for their unconditioned love and inspire my life. Last but least, I would like to thank my dearest wife Elies Hyeri Kim for her understanding and love.

Part of this dissertation is a reprint of published/submitted papers.

Chapter 3, in part, is a reprint of the following publication: Kyoung-Rok Lee, Ramsin Khoshabeh, Truong Q. Nguyen, “Sampling-based Robust Multi-lateral Filtering for Depth Enhancement”, *European Signal Processing Conference*, Bucharest, Romania, Aug. 2012.

Chapter 4, in part, is a reprint of the following publication: Kyoung-Rok Lee, Truong Q. Nguyen, “Robust Tracking and Mapping with a Handheld RGB-D Camera”, *IEEE Winter Conference on Applications of Computer Vision*, Steamboat Spring, CO, Mar. 2014.

Chapter 5, in part, is a reprint of the following publication: Kyoung-Rok Lee, Truong Q. Nguyen, ”Realistic Surface Geometry Reconstruction Using A Handheld RGB-D Camera”, *Machine Vision and Applications*, (submitted), 2014.

Chapter 6, in part, a reprint of the following publication: Kyoung-Rok Lee, Haleh Azartash, Truong Q. Nguyen, ”Accurate, Real-time, and Dense Surface Reconstruction of Dynamic Objects using Multiple RGB-D Cameras”, *Machine Vision and Applications*, (submitted), 2014.

VITA

2008	B. E. in Computer Engineering, Kyungpook National University, South Korea
2010	M. S. in Computer Science (Computer Vision and Graphics), University of California, San Diego
2014	Ph. D. in Electical Engineering (Signal and Image Processing), University of California, San Diego

PUBLICATIONS

Kyoung-Rok Lee, Truong Q. Nguyen, ”Realistic Surface Geometry Reconstruction Using A Handheld RGB-D Camera”, *Machine Vision and Applications*, (submitted), 2014.

Kyoung-Rok Lee, Haleh Azartash, Truong Q. Nguyen, ”Accurate, Real-time, and Dense Surface Reconstruction of Dynamic Objects using Multiple RGB-D Cameras”, *Machine Vision and Applications*, (submitted), 2014.

Haleh Azartash, **Kyoung-Rok Lee** and Truong Q. Nguyen, ”Visual Odometry for Dynamic Environment Using RGB-D Camera with Depth-Aware Image Segmentation”, *IEEE Transaction on Robotics*, (submitted), 2014.

Kyoung-Rok Lee, Truong Q. Nguyen, “Robust Tracking and Mapping with a Handheld RGB-D Camera”, *IEEE Winter Conference on Applications of Computer Vision*, Steamboat Spring, CO, Mar. 2014.

Haleh Azartash, **Kyoung-Rok Lee**, Truong Q. Nguyen, “Visual odometry for RGB-D cameras for dynamic scenes”, *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Florence, Italy, May. 2014.

Kyoung-Rok Lee, Ramsin Khoshabeh, Truong Q. Nguyen, “Sampling-based Robust Multi-lateral Filtering for Depth Enhancement”, *European Signal Processing Conference*, Bucharest, Romania, Aug. 2012.

ABSTRACT OF THE DISSERTATION

Accurate, Efficient, and Robust 3D Reconstruction of Static and Dynamic Objects

by

Kyoung-Rok Lee

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California, San Diego, 2014

Professor Truong Q. Nguyen, Chair

3D reconstruction is the method of creating the shape and appearance of a real scene or objects, given a set of images on the scene. Realistic scene or object reconstruction is essential in many applications such as robotics, computer graphics, Tele-Immersion (TI), and Augmented Reality (AR). This thesis explores accurate, efficient, and robust methods for the 3D reconstruction of static and dynamic objects from RGB-D images.

For accurate 3D reconstruction, the depth maps should have high geometric quality and resolution. However, depth maps are often captured at low-quality or low-resolution, due to either sensor hardware limitations or errors in estimation. A new sampling-based robust multi-lateral filtering method is proposed herein to improve the resolution and quality of depth data. The enhancement is achieved by selecting reliable depth samples from a neighborhood of pixels and applying multi-lateral filtering using colored images that are both high-quality and high-resolution.

Camera pose estimation is one of the most important operations in 3D reconstruction, since any minor error in this process may distort the resulting reconstruction. We present a robust method for camera tracking and surface mapping using a handheld RGB-D camera, which is effective for challenging situations such as during fast camera motion or in geometrically featureless scenes. This is based on the quaternion-based orientation estimation method for initial sparse estimation and a weighted Iterative Closest Point (ICP) method for dense estimation to achieve a better rate of convergence for both the optimization and accuracy of the resulting trajectory.

We present a novel approach for the reconstruction of static object/scene with realistic surface geometry using a handheld RGB-D camera. To obtain high-resolution RGB images, an additional HD camera is attached to the top of a Kinect and is calibrated to reconstruct a 3D model with realistic surface geometry and high-quality color textures. We extend our depth map refinement method by utilizing high frequency information in color images to recover finer-scale surface geometry. In addition, we use our robust camera pose estimation to estimate the orientation of the camera in the global coordinate system accurately.

For the reconstruction of moving objects, a novel dynamic scene reconstruction system using multiple commodity depth cameras is proposed. Instead of using expensive multi-view scene capturing setups, our system only requires four Kinects, which are carefully located to generate full 3D surface models of objects. We introduce a novel depth synthesis method for point cloud densification and noise removal in the depth data. In addition, a new weighting function is presented to overcome the drawbacks of the existing volumetric representation method.

Chapter 1

Introduction

1.1 Problem Statement

3D reconstruction is the process of generating the geometry and appearance of real objects using computer vision and computer graphics. Building realistic 3D models of real environments is an important task in computer vision. There has been growing interest in the reconstruction of realistic 3D models of objects or scenes using stereo cameras [2], range sensors [3, 4], and uncalibrated image collections [5, 6]. Some methods produce very accurate, high-quality results; however, image-based methods suffer from problems of noise contamination and the presence of illumination changes in different views. Laser scanning or structured light systems may require specialized training, expensive devices, or special environmental setups, all of which restrict their ability to be commonly used in daily life.

Recently, affordable RGB-D cameras such as Intel's Interactive Gesture Camera and Microsofts Kinect [7] have been rapidly adopted by computer vision researchers. These low-price RGB-D sensing technologies have enabled many system capabilities for both customers and researchers. RGB-D cameras have several advantages over other capturing devices; first, an RGB-D camera can be operated in the same way as ordinary video cameras, and so it is easy to use and requires no specialized training. Second, since RGB-D sensors use infrared light for depth measurement, it is independent from scene color textures and does not interfere with the visual information of the scene. Therefore, we are able to utilize both color and depth information for reconstruction. Finally, they can measure 3D information of the scene at the rate of a video, which enables real-time surface model generation. In this thesis, we present 3D capturing systems that generate 3D reconstructions of both stationary and dynamic scenes using RGB-D images.

1.2 Motivation

The development of real-time 3D scanning technology with low cost devices has allowed researchers and consumers to digitalize high quality models of objects or scenes easily. The applications of 3D reconstruction are diverse, and include human body scanning system, geometry-aware scene interaction, view synthesis for multi-view display, and Archeological research/education.

- *Real-time human body scanning system*: Real-time reconstruction will provide a 3D representation of the user's body. The system can be used for the treatment of patients with neurological disorders by displaying a realistic, allocentric view of the body to manipulate visual body feedback. The advantages of this technology lie in its rapid, non-intrusive, high-precision data capture, which can be used to build personalized 3D models. These models can be used in various other applications, including anthropometry, animation, and consumer-retail situations.

- *Geometry-aware scene interaction*: With a detailed 3D model, user interaction can be performed in the reconstructed scene. A user's touch will be determined by computing the intersection between their body and background surfaces; the visual feedback from touch creates an immersive interaction experience for the user.

- *View synthesis for multi-view display*: Over the past few years, there has been increasing interest in 3D display research. The ability to synthesize multiple views is essential for autostereoscopic displays, which is glasses-free 3D visualization. Given a dense 3D reconstruction, a series of multiple plausible virtual views along the stereoscopic baseline can be generated from multiple virtual cameras for presentation in autostereoscopic displays.

- *Archeological research/education*: A digitalized 3D model of real objects can be used for research and education in Archeological study. 3D scanning of objects and specific locations enables innovative applications for the historical archive. The virtual archive will be a good way for both scholars and the public to access and interact with historical objects or sites.

1.3 Contributions

This work presents a novel depth map enhancement method, robust camera pose estimation, realistic stationary object reconstruction using a handheld RGB-D camera, and dynamic scene reconstruction using multiple RGB-D cameras. Our main contributions are:

1. We present a new method of enhancing noisy or low-resolution depth maps using high-resolution color images. Our method is based on sample selection and refinement in conjunction with multi-lateral filtering.
2. The proposed robust camera pose estimation method is effective in challenging situations such as during fast camera motion or in geometrically featureless scenes. Our approach is based on sparse visual feature-based tracking in conjunction with dense estimation using a weighted ICP method.
3. We propose an accurate and robust 3D reconstruction system which utilizes high quality RGB images by attaching a HD RGB camera onto a Kinect to reconstruct a 3D model with realistic surface geometry and high-quality color textures.
4. For accurate dynamic scene reconstruction, we present a depth synthesis technique that generates synthesized depth maps at virtual viewpoints to densify the point cloud so that it covers some surfaces that may be inaccessible to the cameras, and a reliability-based weight function to overcome the drawback of the existing volumetric representation method.

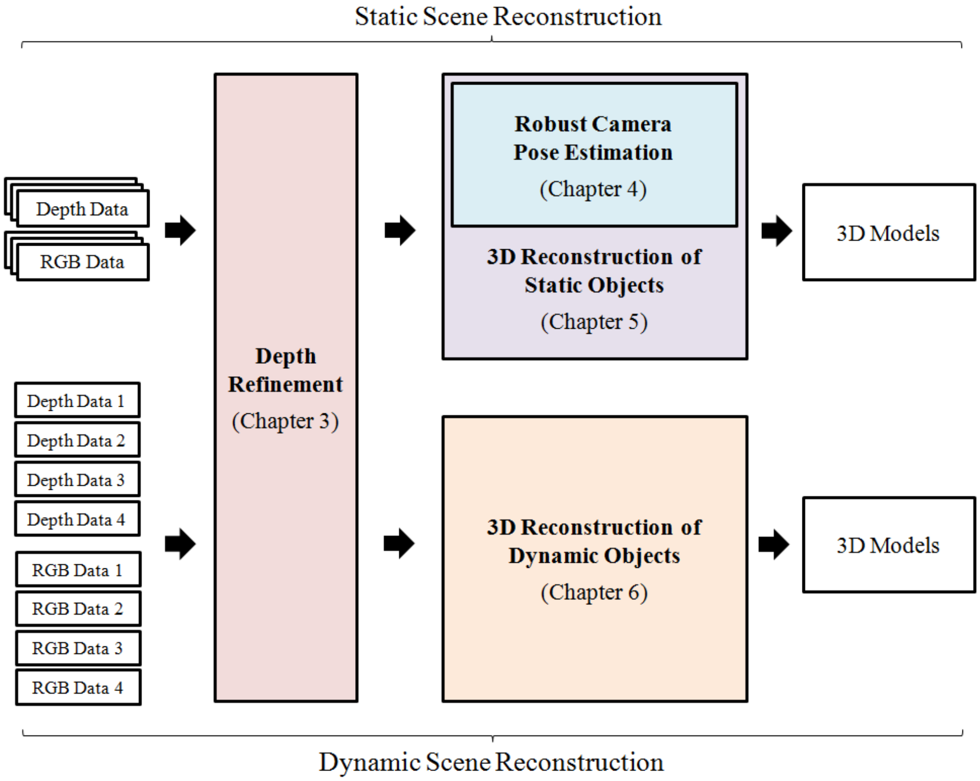

Static Scene Reconstruction
Depth Data
RGB Data
Depth Data 1
Depth Data 2
Depth Data 3
Depth Data 4
RGB Data 1
RGB Data 2
RGB Data 3
RGB Data 4
Depth Refinement (Chapter 3)
Robust Camera Pose Estimation (Chapter 4)
3D Reconstruction of Static Objects (Chapter 5)
3D Reconstruction of Dynamic Objects (Chapter 6)
3D Models
3D Models
Dynamic Scene Reconstruction

Figure 1.1: A summary of the structure of the dissertation

1.4 Organization

Figure 1.1 shows the organization of this dissertation. This dissertation is organized as follows:

Chapter 2: This chapter provides the background information of this dissertation. The basic pinhole camera model and projective geometry for the Kinect depth map are reviewed. In addition, the specifications and mechanisms of the Kinect, the device used in this work, are presented.

Chapter 3: This chapter explores a novel depth refinement method. It presents sample selection and multi-lateral filtering techniques to enhance the quality of raw depth data. Evaluations on the Middlebury dataset and performance gains over existing methods are provided.

Chapter 4: In this chapter, a robust camera pose estimation method for accurate trajectory estimation is investigated. Quantitative results on an RGB-D trajectory benchmark dataset is provided.

Chapter 5: This chapter presents a novel method for the reconstruction of real objects/scenes with realistic surface geometry using a handheld RGB-D camera. Comparisons with other 3D reconstruction methods using real-life datasets are provided.

Chapter 6: This chapter presents a real-time 3D reconstruction method that generates an accurate surface model of dynamic scenes using multiple Kinect cameras. Metrical evaluations and computational analysis on various real-life datasets are provided.

Chapter 7: This chapter presents the conclusions of the dissertation and reviews the contributions; future research directions will be discussed.

Chapter 2

Background

This chapter describes background information and concepts used within this dissertation. Section 2.1 introduces the recent RGB-D sensing device, Kinect. Section 2.2 briefly covers the basic pinhole camera model and camera projection matrix for depth map generated by Kinect.

2.1 Kinect

Many researchers have been extensively studying the subject of 3D reconstruction with range sensing devices including structured light camera, and laser scans. However, these devices are expensive and require highly specialized knowledge and practice to use them. Kinect is a recently developed RGB-D sensing device, and is originally designed for natural human interaction in gaming environment. Since it produces generally good depth map in real-time at an affordable cost, it is widely used in computer vision applications.

Figure 2.1 shows the components of the Kinect device. The Kinect sensor consists of an RGB camera, an infrared (IR) pattern projector, and an IR camera. For purposes of sensing 3D information, the IR projector emits a predefined pattern of speckles onto the scene and the IR camera captures the scene to triangulate 3D scene [7]. In Figure 2.2, the distance between a_i and a_{i+1} is defined in the reference and the distance between b_j and b_{j+1} is the observed distance. The sensor knows the calibration information between the IR projector and the IR camera and the distances between points of the

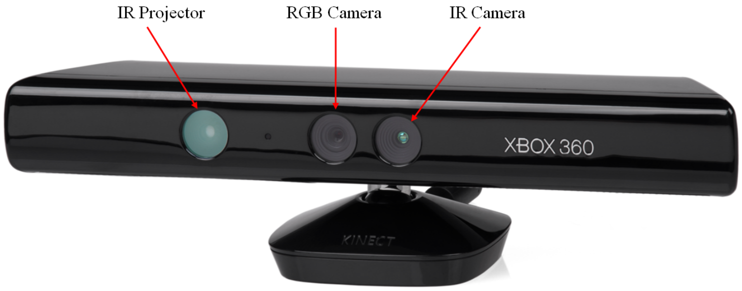


Figure 2.1: Components of Kinect.

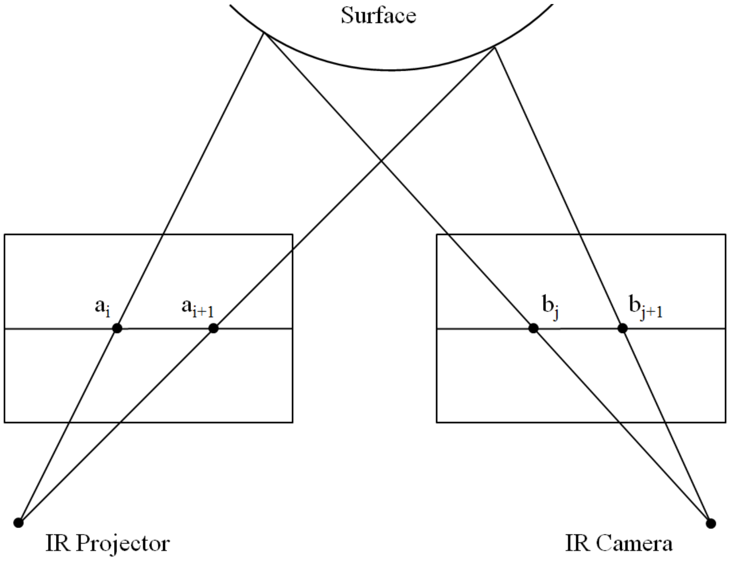
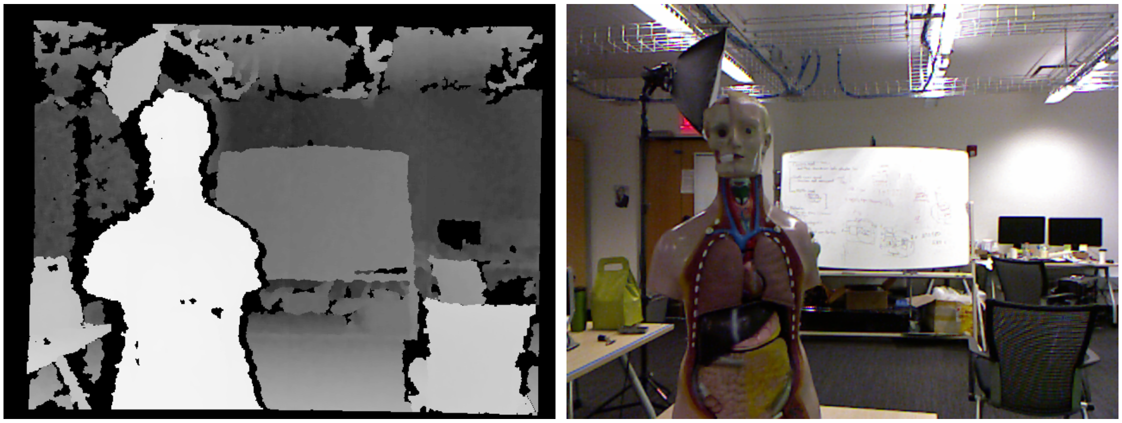


Figure 2.2: Triangulation of each speckle between a predefined pattern and observed pattern.



(a) Depth map

(b) Color image

Figure 2.3: Kinect data. The calibration is performed in depth map to align with color image. (a) Calibrated depth map. (b) Color image. In (a), nearer surfaces are brighter; further surfaces are darker. The black pixels represent the occlusion region.

pattern at reference distance. Therefore, the depth is computed by triangulation of each speckle between the expected pattern at reference distance and the observed pattern; if the distance of points is larger than the predefined distance, it means that the object is closer; further surfaces would have smaller distance. This distortion of IR pattern allows for computing the 3D structure of the scene. The Kinect captures 640×480 depth data and 640×480 color data at 30 fps. The range of operation is from $0.7m$ to $6m$ approximately.

Although Kinect has many advantages over other existing range sensing methods, it also has some drawbacks. Since the speckle pattern from the IR transmitter in Kinect cannot cover detail, it causes occlusion regions as well as errors in the boundary area. Figure 2.3 shows the calibrated depth and the color image captured by Kinect. In the depth map, the black pixels in depth map represent unknown region where Kinect cannot obtain depth information. The possible reasons for this are: the region is occluded from the point of view of the IR camera, or infrared light is absorbed in that region.

2.2 Camera Model and Calibration

For understanding 3D reconstruction problem, the first step is to understand camera model and projective geometry. In this section, we review the pinhole camera model and projective geometry for depth map from Kinect.

2.2.1 Basic Pinhole Camera Model

In the basic pinhole camera model, 3D points in camera space are projected onto an image plane placed at the position $z = f$. Figure 2.4 shows a camera with camera center $\mathbf{O}$ and the principal axis parallel to the Z axis.

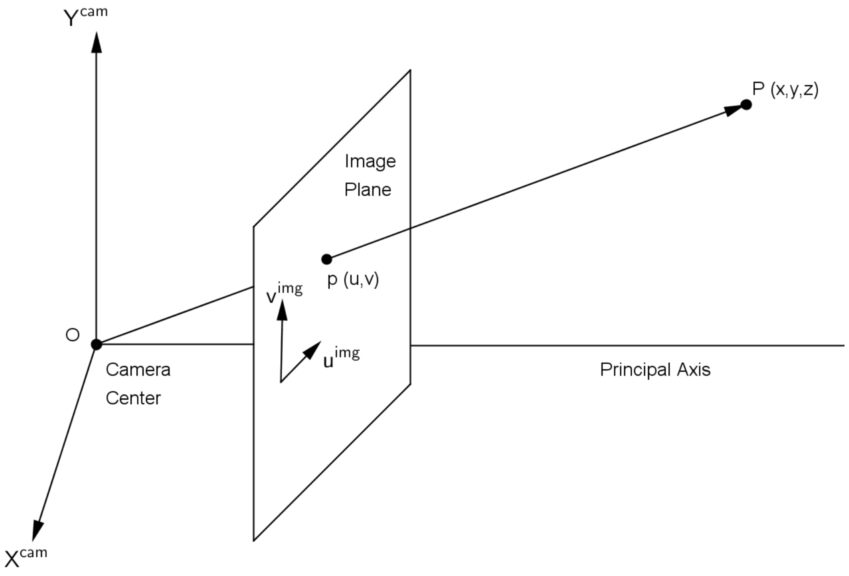


Figure 2.4: Basic pinhole camera model.

A 3D point $\mathbf{P} = [\, x \;\; y \;\; z \,]^{\top}$ is projected on the camera's image plane at position $\mathbf{p} : [\, u \;\; v \,]^{\top}$. By using ”similar triangles” (the sides facing the equal angles are always in the same ratio), the position of the point on the image plane $\mathbf{p}$ is computed as:

$$\frac{f}{z} = \frac{u}{x} = \frac{v}{y} \tag{2.1}$$

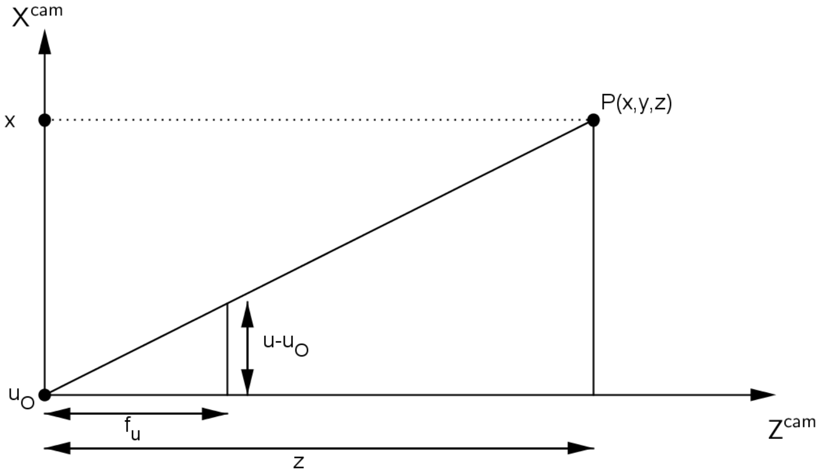


Figure 2.5: Similar triangles for perspective camera

which gives us:

$$u = \frac{fx}{z} \tag{2.2}$$

$$v = \frac{fy}{z} \tag{2.3}$$

If the 3D and image points are represented by homogeneous vectors, we can rewrite this as:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.4}$$

We can verify that this generates $u = \frac{fx}{z}$ and $v = \frac{fy}{z}$ when we simply divide them by the third coordinate w to transform a point in the projective plane back into Euclidean coordinates.

2.2.2 Projective Geometry for Depth Map from Kinect

While the basic pinhole camera model assumes image coordinates are Euclidean coordinates, the image plane of Kinect cameras is formed by pixels. Therefore, we need to convert the Euclidean coordinates to image pixel coordinates. For this operation, we first need to know the resolutions of the camera in pixels/millimeters since $\mathbf{p}$ should be expressed in pixels. We use r_u and r_v to represent the resolution of the camera in terms of pixel dimensions on the u and v directions, respectively. Also, the origin of the 2D image coordinate system can be different from where the Z axis intersects the image plane. We define this translation by (t_u, t_v). As a result, the pixel coordinates from Euclidean 3D points are given as

$$u = r_u \frac{fx}{z} + r_u t_u \tag{2.5}$$

$$v = r_v \frac{fy}{z} + r_v t_v \tag{2.6}$$

We can express the above in matrix form as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} r_u f & 0 & r_u t_u \\ 0 & r_v f & r_v t_u \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underbrace{\begin{bmatrix} f_u & 0 & u_O \\ 0 & f_v & v_O \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{P}} = \mathbf{KP} \tag{2.7}$$

where f_u, f_v are focal lengths in pixels along u and v axis respectively, u_O and v_O are translations in pixels along u and v axis respectively. We call the upper triangular matrix $\mathbf{K}$ as the camera calibration matrix or intrinsic parameter matrix for the camera. Figure 2.5 shows similar triangles for Kinect camera.

Chapter 3

A Depth Refinement Method

Depth maps are commonly used in many 3D applications. For these applications, the depth maps should be of high geometric quality and resolution since a minor error may result in distortions. Recent advances use various types of sensors to obtain depth maps, such as Time-of-Flight cameras(ToF) [8], real-time infrared projectors and cameras (e.g. Microsoft Kinect), or stereo vision systems. Unfortunately, most of the time, the quality and resolution of the acquired depth is not up to par with the analogous color images obtained from standard cameras.

Due to this limitation, the subject of depth map up-sampling/refinement has been extensively studied. Many existing methods try to solve the problem by aligning the low-resolution depth map with its corresponding high-resolution color image. They obtain up-sampled depth maps by taking weighted averages of depth values in a local window for each pixel. Therefore, the window size is the most critical parameter affecting performance, which is undesirable.

In this chapter, we introduce a novel method to enhance noisy or low-resolution depth maps using high-resolution color images. Our approach is based on selecting reliable depth samples from a neighborhood of pixels and applying multi-lateral filtering. Figure 3.1 shows the flow chart of our method. We first define unreliable regions by calculating a measure of reliability for each pixel in the depth map. The reliability is determined by calculating the sum of gradients for each pixel's neighborhood. Every pixel in the region of low reliability collects samples from the region of high reliability and selects the best sample with the highest fidelity. Each pixel's selected depth sample

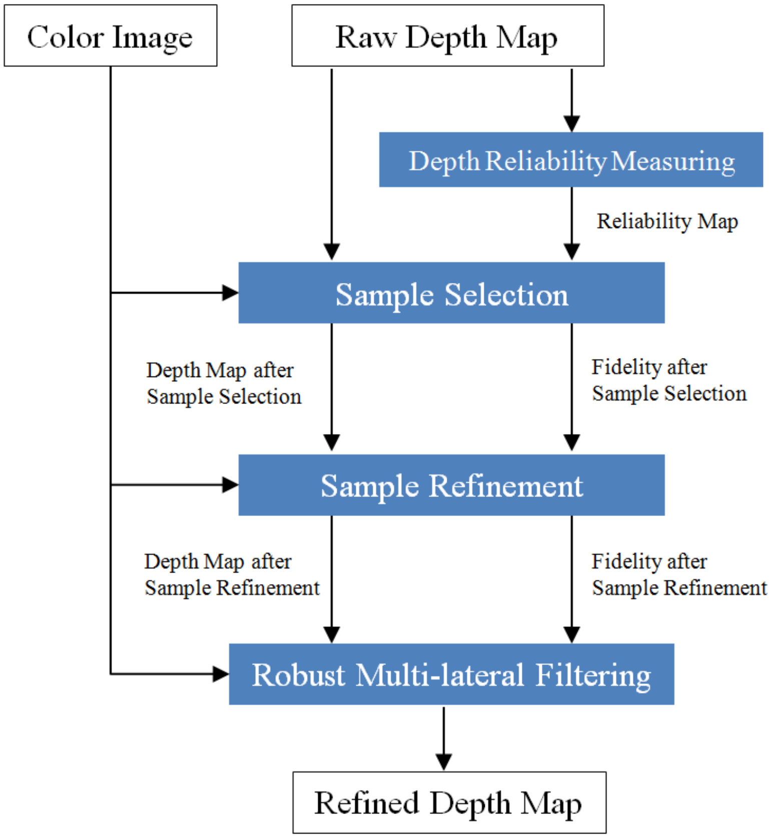

Color Image
Raw Depth Map
Depth Reliability Measuring
Reliability Map
Sample Selection
Depth Map after Sample Selection
Fidelity after Sample Selection
Sample Refinement
Depth Map after Sample Refinement
Fidelity after Sample Refinement
Robust Multi-lateral Filtering
Refined Depth Map

Figure 3.1: Flow chart of the proposed depth refinement method.

is refined by sharing its information with its neighbors' selected samples in the sample refinement stage. Finally, a robust multi-lateral filter, which is an extended joint bilateral filtering technique with an additional factor for robustness weights, is applied to reduce noise while preserving sharpness along edges. We evaluate our approach on the Middlebury dataset [9] and show that our method provides performance gains over existing methods.

The remainder of this chapter is organized as follows. Section 3.1 describes the related work. Section 3.2 presents our proposed method. A visual and quantitative comparison of a number of key methods and improvement results are reported in Section 3.3. Finally, a summary is given in Section 3.4.

3.1 Related Work

As depth sensors have been widely used in the field of computer vision, the problem of depth refinement has received ever-increasing attention. A seminal work in the study of the depth map refinement problem is the work by Diebel et al. [10]. They assumed that discontinuities in range and color tend to co-align. In their work, the posterior probability of the high-resolution reconstruction is designed as a Markov Random Field (MRF) and it is optimized with the Conjugate Gradient (CG) algorithm.

Following with a similar depth refinement method, Kopf et al. proposed Joint Bilateral up-sampling (JBU) [11]. This approach leverages a modified bilateral filter. The traditional bilateral filter is an edge-preserving filter where the value of each output pixel is a weighted average of pixels in a neighborhood [12]. They up-sample a low-resolution depth D by applying a spatial filter, while jointly apply a similar range filter on the high-resolution color image I. In other words, the up-sampled depth value at pixel $\mathbf{p}$ is determined by:

$$D(\mathbf{p})^{JBU} = \frac{\sum_{\mathbf{q}\in\Omega_{\mathbf{p}}} f_r(|I(\mathbf{p}) - I(\mathbf{q})|) f_s(||\mathbf{p}-\mathbf{q}||) D(\mathbf{q})}{\sum_{\mathbf{q}\in\Omega_{\mathbf{p}}} f_r(|I(\mathbf{p}) - I(\mathbf{q})|) f_s(||\mathbf{p}-\mathbf{q}||)} \tag{3.1}$$

where $\Omega_{\mathbf{p}}$ is the support window centered at $\mathbf{p}$. Here, f_r is the range filter and f_s is the spatial filter, which are Gaussian filters with variances σ_r and σ_s, respectfully. Research in depth map refinement has recently experienced large progress by the initial introduction of JBU [13, 14, 15, 16, 17, 18, 19, 20, 21].

Additionally, Yang et al. presented an refinement method based on bilateral filtering the cost volume with sub-pixel estimation [22]. They build a cost volume of depth probability and then iteratively apply a standard bilateral filter to it. The final output depth map is generated by taking the winner-takes-all approach on the weighted cost volume. Finally, a sub-pixel estimation algorithm is applied to reduce discontinuities.

It is worth noting that all of the methods mentioned above may suffer from artifacts, such as texture copying and edge blurring. Texture copying occurs in smooth areas with noisy depth data and textures in the color image, while edge blurring occurs in transition areas if different objects (located in different depth layers) have similar color.

To overcome these problems, the Noise Aware Filter for Depth Up-sampling (NAFDU) was proposed by [14]. The method switches between two different filters; a standard bilateral up-sampling filter on smooth regions and a joint bilateral up-sampling filter on transition areas in the depth map. A blending function is used for a gradual intermixing between the two filter outputs.

The approach closest to ours, the Pixel Weighted Average Strategy (PWAS) by Frederic et al., also proposes to resolve artifacts [17]. They build multi-lateral up-sampling filters, which are an extended joint bilateral filter with an added credibility factor. The factor takes into account the low reliability of depth measurements along depth edges and the inherent noisy nature of real-time depth data. As a further improvement upon PWAS, Adaptive Multi-lateral Filtering (AMF) has been proposed to improve accuracy within smooth regions [19].

These approaches may solve the texture copying and edge blurring problems, but their performances are unfortunately very sensitive to the window size of their filter, making the window size the most critical parameter. If the window size is too large, it might cause boundary blurring and lose details of complex objects. If the window size is too small, it may fail to collect significant information from its neighborhood.

In light of these problems, we present a new depth sampling method and multilateral filtering technique. Our method is based on the concept of sample selection and refinement, introduced by [23], and multi-lateral filtering, which is inspired by [17]. To our knowledge, our work is the first of its kind that presents a sampling-based depth refinement.

3.2 Proposed Method

Errors in range images generated by real-time depth sensors or stereo vision systems can be roughly categorized into two broad categories:

- Errors in transition areas: Inadequate calibration, occlusion area, or motion artifacts often lead to wrong distance values at object boundaries when we fuse the depth maps with color images.
- Random noise on geometrically flat or smooth surfaces: Properties of the object surface, lighting conditions, or systematic errors may generate noise on the surface.

In our work, we investigate a method that is able to fix both errors. Our method takes a color image I and a depth map D as inputs. The process consists of sample selection, sample refinement, and robust multi-lateral filtering. Before refining the depth map, we first measure depth reliabilities and define unreliable regions in it. In the sample selection stage, for every pixel in the unreliable region we collect samples from reliable regions and select the best sample giving the highest fidelity. Then the selected depth samples are refined by sharing their information with their neighbors' selected samples. Finally, a robust multi-lateral filter is applied to reduce noise in smooth areas, while preserving sharpness along the edges.

3.2.1 Unreliable Region Detection

We use the gradient of the depth map as an important key for measuring reliability of depth values based on our assumption that depth values with high variance in their neighborhood or depth values along edges are not reliable. We first take the derivative of the depth map and calculate its magnitude. Then for each pixel, we compute an average function of Gaussian of the gradient magnitude in a small window. Thus, the reliability

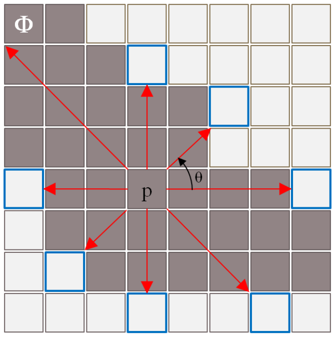


Figure 3.2: Pixel $\mathbf{p}$ shoots several rays (red lines) toward reliable (white) region and collects the closest samples (blue squares).

for each pixel is determined by the following equation:

$$R(\mathbf{p}) = \sum_{\mathbf{q}\in\Omega_{\mathbf{p}}} f_t(|D(\mathbf{p}) - D(\mathbf{q})|)/|\Omega\mathbf{p}| \tag{3.2}$$

where f_t is the Gaussian function with variance σ_t , and $\Omega_{\mathbf{p}}$ is the window centered at pixel $\mathbf{p}$. The unreliable region Φ is the set of pixels whose reliability values are less than a certain threshold:

$$\Phi \leftarrow \{\mathbf{p}|R(\mathbf{p}) < \tau\} \tag{3.3}$$

The threshold τ controls the width of the unreliable region. The remaining pixels in the depth map are identified as the reliable region. Every depth value in this unreliable region Φ will be refined by the following sample selection and refinement steps.

3.2.2 Sample Selection

To recover the pixels in the unreliable region, we collect depth samples from a nearby reliable region, inspired by [23]. In the alpha matting problem, color samples are extracted from the reliable regions to estimate the unknown pixels' alpha values. Similar to this, we collect depth samples from the reliable regions to refine the unreliable depth values.

We collect depth samples from a neighborhood by shooting several rays toward the known region. The slope of each ray is given by θ, $\theta \leftarrow \{\frac{j\pi}{\eta}|j = 0, 1, ..., 2\eta - 1\}$. When the ray from pixel $\mathbf{p}$ meets the known region, the closest depth sample is saved to

$\Psi_{\mathbf{p}}$. Figure 3.2 illustrates rays (red lines) and collected depth samples (blue squares) for the pixel location $\mathbf{p}$.

The best depth sample for pixel $\mathbf{p} \in \Phi$ is obtained by calculating the fidelity for all collected depth samples $\Psi_{\mathbf{p}}$. For every $\mathbf{i} \in \Psi_{\mathbf{p}}$, the fidelity function is computed based on the criteria that pixels with similar color tend to share similar depth values and that they are likely to have the same depth value if they are spatially close. Therefore, at a given pixel $\mathbf{p}$, we compute the fidelity function as:

$$g^{SS}(\mathbf{p}, \mathbf{i}) = \sum_{\mathbf{q} \in \Omega_{\mathbf{p}}} f_r(|I(\mathbf{q}) - I(\mathbf{i})|) f_s(||\mathbf{q} - \mathbf{i}||)/|\Omega \mathbf{p}| \tag{3.4}$$

where f_r and f_s are taken to be Gaussian functions with standard deviations σ_r and σ_s respectively. The chromatic similarity in the RGB color space is computed by Euclidean distance metric. To suppress errors from image noise caused by low lighting, high ISO settings, or chromatic distortion, we take into account the average of all pixels in a 3×3 window centered at pixel $\mathbf{p}$.

Eq. (3.4) will have a large response for the depth value $D(\mathbf{i})$ which has a similar color and spatial proximity, but a value close to zero for the rest. We select the $\mathbf{i}^*$ giving the largest fidelity value among depth samples:

$$\mathbf{i}^* = \underset{\mathbf{i} \in \Psi_{\mathbf{p}}}{\arg\max}\, g^{SS}(\mathbf{p}, \mathbf{i}) \tag{3.5}$$

Then we save the selected depth and fidelity for each pixel $\mathbf{p} \in \Phi$.

$$\begin{aligned} D^{SS}(\mathbf{p}) &= D(\mathbf{i}^*), \\ E^{SS}(\mathbf{p}) &= g^{SS}(\mathbf{p}, \mathbf{i}^*) \end{aligned} \tag{3.6}$$

We use the selected disparity map D^{SS} and the fidelity map E^{SS} in the following sample refinement stage.

3.2.3 Sample Refinement

Since we collect depth samples with the ray searching scheme, where several rays spread out like the spokes of a bicycle wheel, the method occasionally fails to collect appropriate depths in some situations. For example, the desirable depth sample may be located between the rays or it may happen that the color affected by noise with a false depth sample is accidentally very similar to the target's color. Therefore, an additional refinement stage is required.

In the sample refinement stage, the samples are refined by comparing their own choice of best depth sample with the choices of their neighborhood. The design of the fidelity function for sample refinement is based on color fitness and spatial distance between pixel $\mathbf{p}$ and its neighbor $\mathbf{q}$ as well as $\mathbf{q}$'s fidelity value from the sample selection stage. The fidelity for sample refinement is determined by:

$$g^{SR}(\mathbf{p},\mathbf{q}) = E(\mathbf{q})^{SS} f_r(|I(\mathbf{p}) - I(\mathbf{q})|) f_s(||\mathbf{p}-\mathbf{q}||) \tag{3.7}$$

where $E(\mathbf{q})^{SS}$ is the fidelity value of the pixel $\mathbf{q}$ from the previous stage. $\mathbf{q}^*$ is chosen to give the largest fidelity value among $\mathbf{p}$'s neighborhood:

$$\mathbf{q}^* = \arg\max_{\mathbf{q}\in\Omega_{\mathbf{p}}} g^{SR}(\mathbf{p},\mathbf{q}) \tag{3.8}$$

Similar to the sample selection stage, both refined depth value and fidelity are saved.

$$\begin{aligned} D^{SR}(\mathbf{p}) &= D)(\mathbf{q}^*), \\ E^{SR}(\mathbf{p}) &= g^{SR}(\mathbf{p},\mathbf{q}^*) \end{aligned} \tag{3.9}$$

Up to this point, we have estimated new depth values for pixels in the unreliable region. In the next step, the refined depth data will be used as an initial depth estimate and the fidelity values will be used to determine the robustness factor.

3.2.4 Robust Multi-lateral Filtering

After the unreliable region is refined by the sample selection and refinement stages, the depth map still needs to be processed to reduce discontinuities in the final depth map. Therefore, we apply a robust multi-lateral filter. It is an extended joint bilateral filtering technique, but the robustness factor is added to reduce blurring along edges as well as to refine edges. The robustness value for pixel $\mathbf{p}$ is determined by choosing the minimum value between $R(\mathbf{p})$ and $E(\mathbf{p})^{SR}$ since we want to disregard depth values with low plausibility as much as we can:

$$\tilde{R}(\mathbf{p}) = \min\{R(\mathbf{p}), E(\mathbf{p})^{SR}\} \tag{3.10}$$

With this robustness factor, our final depth is determined by

$$D(\mathbf{p})^{MF} = \frac{\sum_{\mathbf{q}\in\Omega_{\mathbf{p}}} f_r(|I(\mathbf{p}) - I(\mathbf{q})|) f_s(||\mathbf{p} - \mathbf{q}||)\tilde{R}(\mathbf{q})D(\mathbf{q})^{SR}}{\sum_{\mathbf{q}\in\Omega_{\mathbf{p}}} f_r(|I(\mathbf{p}) - I(\mathbf{q})|) f_s(||\mathbf{p} - \mathbf{q}||)\tilde{R}(\mathbf{q})} \tag{3.11}$$

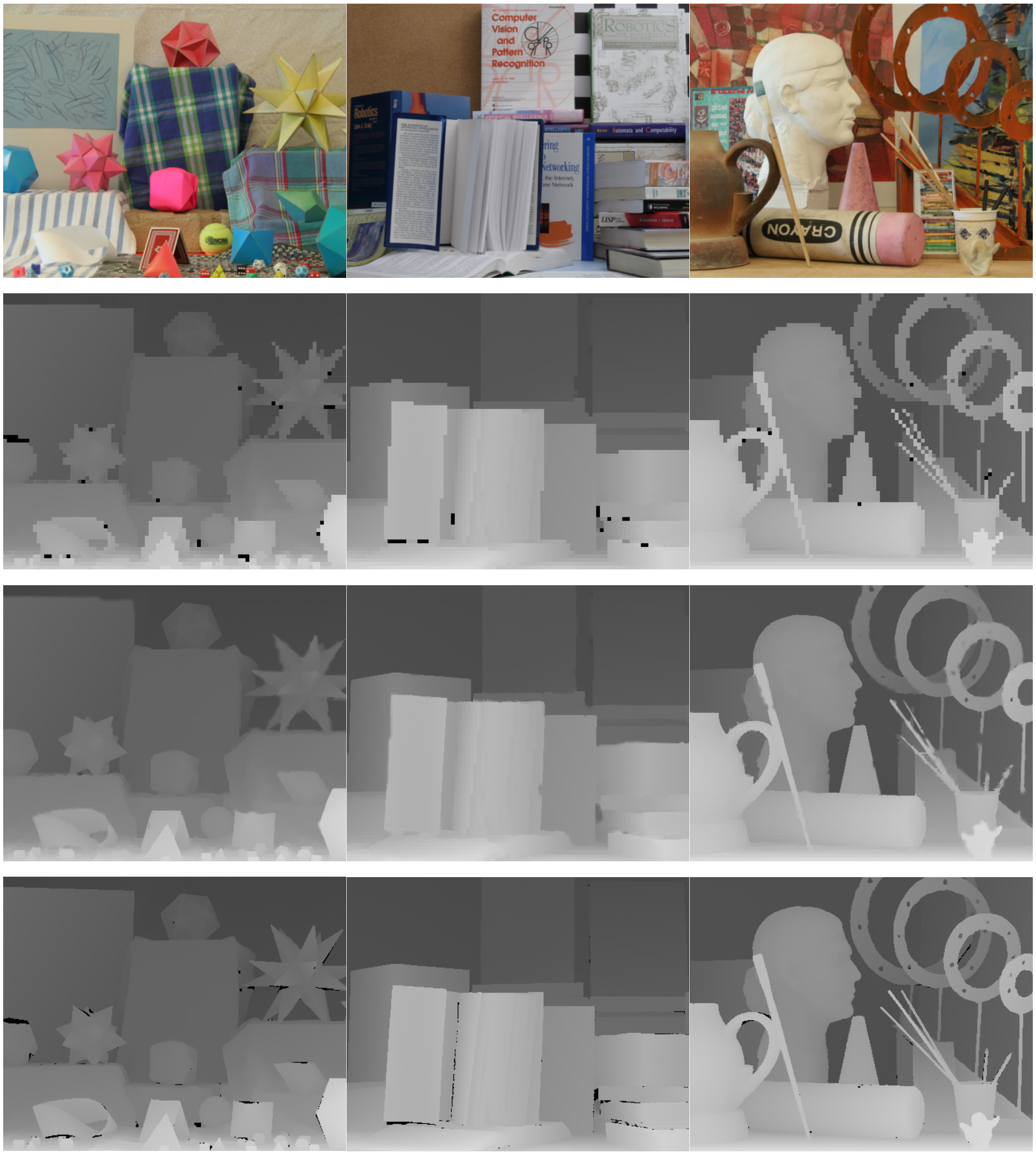
The spatial weighing term f_s is based on pixel position and the range weight f_r is based on color data. Thus, this filter adjusts the edges in the input depth map D^{SR} to the edges in the guidance color image I and the robustness factor $\tilde{R}$ gives low weight to depth values with low fidelity, preventing artifacts such as texture copying and edge blurring. Figure 3.3 shows some of the results of our algorithm as will be discussed in the next section.

3.3 Results

In this section we describe the experiments performed to evaluate our algorithm. We provide both qualitative and quantitative comparisons with other existing methods. Also, we show an improvement benchmark by applying our refinement method to disparity estimation results from all of the 109 methods on the Middlebury stereo evaluation website.

3.3.1 Visual and Quantitative Comparison

For a quantitative comparison, we utilize the *Moebius*, *Books*, and *Art* scenes from the Middlebury datasets. We have evaluated the performance of the proposed method against the state-of-the-art methods presented by [19]. Down-sampled disparity maps are generated by down-sampling the ground truth by a factor of $3\times$, $5\times$, and $9\times$. In [19], they used the structural similarity (SSIM) measure as a quantitative comparison. SSIM is a method for measuring perceptual image quality [24]. However, this measure is not appropriate for depth map evaluation because it does not function properly with disparities in unknown or occluded regions. Middlebury's ground truth maps contain regions of unknown disparity and depth refinement algorithms do not produce meaningful results in those regions. As a fair comparison, we calculate the average percentage of bad pixels with an error threshold of 1 for all known regions; pixels whose disparity



Moebius *Book* *Art*

Figure 3.3: Experimental results of our method. The input disparity maps are generated by downsampling ground truth with factor of 5. (First Row) Color images, (Second Row) Input disparity map, (Third Row) Our results, and (Fourth Row) Ground truths

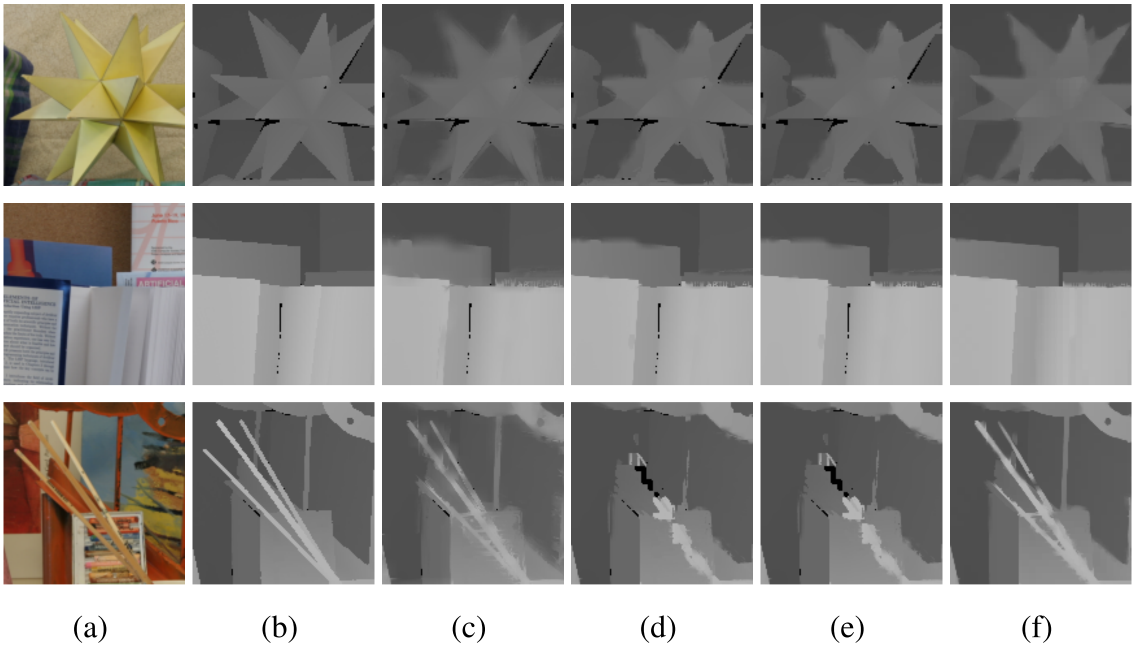


Figure 3.4: Visual comparison on the Middlebury dataset. (a) Color image, (b) Ground truth, The refinement methods include: (c) JBU, (d) PWAS, (e) UML, (f) proposed method

error is greater than threshold are regarded as the bad pixels. This is the same scoring scheme employed in the Middlebury evaluation. Figure 3.4 and Table 3.1 show that our method performs better than all of the other methods.

3.3.2 Improvements

Also, we apply our refinement method to the disparity estimation results of all methods submitted to the Middlebury stereo evaluation. Figure 3.5 shows the improvement in terms of the percentage of bad pixels. Note that the proposed method improves the results of most methods. One limitation of the proposed algorithm is that its performs drops with small and complex images or poorly estimated initial disparity maps.

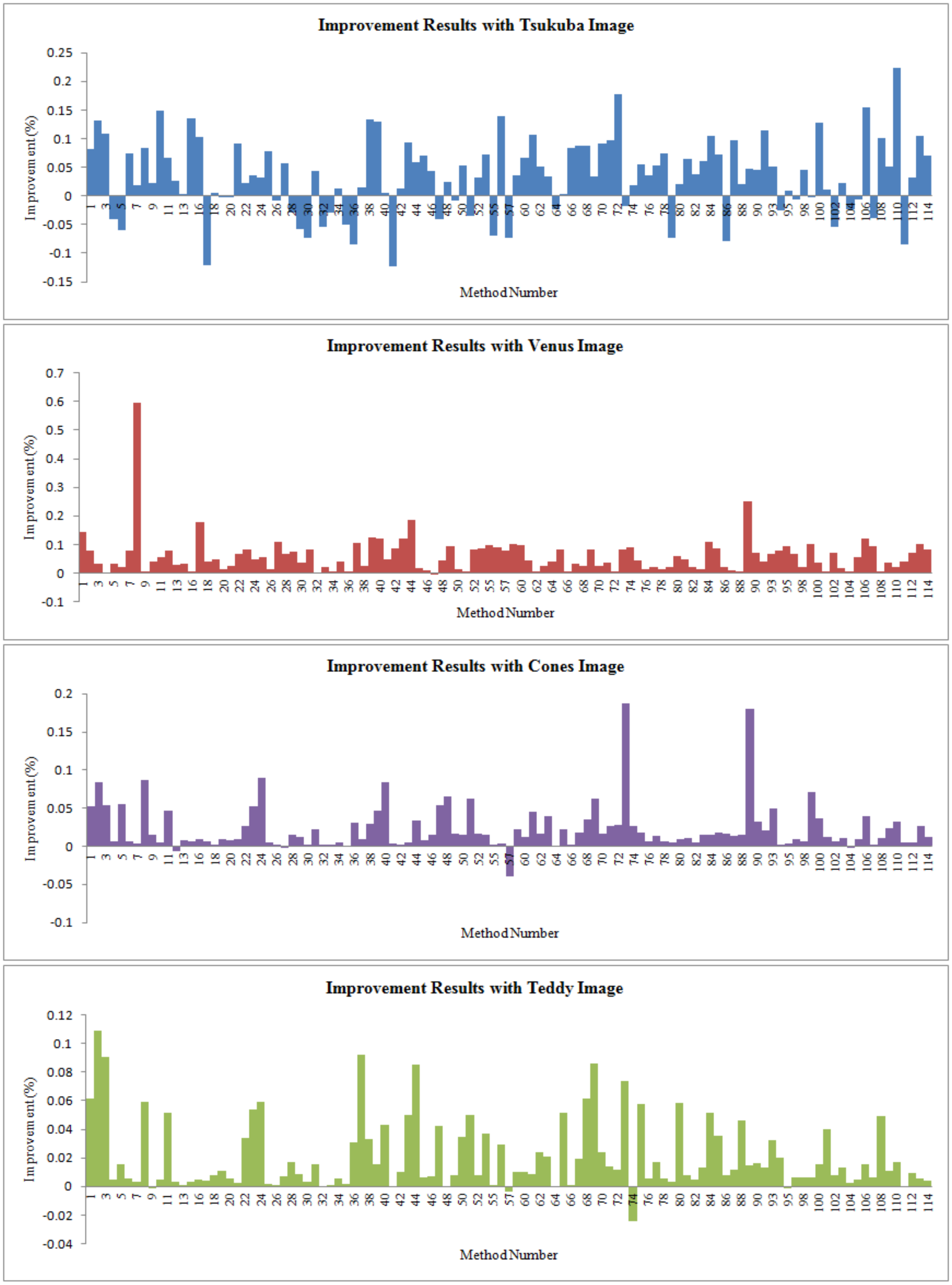


Figure 3.5: Percentage improvement in terms of number of bad pixels after applying the proposed algorithm to all the 109 methods on the Middlebury stereo evaluation.

Table 3.1: Quantitative comparisons (average percent of bad pixels)

Dataset		JBU	PWAS	AMF	Proposed Method
Moebius	3x	7.43	4.68	4.5	**3.62**
	5x	12.22	7.49	7.37	**4.87**
	9x	21.02	12.86	12.75	**9.02**
Books	3x	5.4	3.59	3.48	**2.38**
	5x	9.11	6.39	6.28	**3.58**
	9x	15.85	12.39	12.24	**7.11**
Art	3x	15.15	7.05	6.79	**5.07**
	5x	23.46	10.35	9.86	**6.91**
	9x	38.41	16.87	16.87	**11.7**

3.4 Summary

In this chapter, we have proposed a new depth map enhancement method based on sample selection, refinement, and robust multi-lateral filtering. Specifically, we have introduced a new sampling method to get better accuracy on boundaries. Also we have investigated multi-lateral filtering with a new robustness factor. Experiments clearly show that our algorithm significantly outperforms other state-of-the-art methods in the depth refinement problem.

3.5 Acknowledgments

This chapter is in part a reprint of the following publication: Kyoung-Rok Lee, Ramsin Khoshabeh, Truong Q. Nguyen, “Sampling-based Robust Multi-lateral Filtering for Depth Enhancement”, *European Signal Processing Conference*, Bucharest, Romania, Aug. 2012.

Chapter 4

A Robust Camera Pose Estimation Method

Camera pose estimation is the process of recovering the relative orientation and position of one camera system to another. In recent years, there is an increasing interest in camera localization problem due to the fact that it is a fundamental operation in many computer vision applications such as robot localization, 3D reconstruction, and augmented reality.

Many studies of sparse feature-based approaches using visual information have been proposed [25]. Parallel Tracking and Mapping (PTAM) system is proposed in [26], which achieves remarkable accuracy and real-time interactivity in camera tracking. However, the sparse feature-based approach has several drawbacks. Motion blur from rapid camera motion causes erroneous feature localization which in turn results in tracking failure. Besides, there is a possibility that valuable information in images is ignored since only a limited number of points are used in the estimation.

With the advance of low-cost depth sensing devices such as time-of-flight (ToF) and Microsoft's Kinect, numerous studies exploiting both visual and per-pixel depth information have achieved great progress in Simultaneous Localization And Mapping (SLAM) in recent years [1, 27, 28, 29, 30, 31, 32]. While these methods are promising for some datasets, serious challenge arises in situations where camera motion is fast or the scene has very few geometric features.

In this chapter, we propose a robust method for camera tracking and surface mapping using a handheld RGB-D camera. Our method is based on a sparse visual feature-based tracking in conjunction with a dense estimation using a weighted ICP method. The proposed algorithm first performs an initial motion estimation using sparse feature detection and matching which assists in reducing the negative effects of rapid camera motion. Then, dense estimation is applied by weighted ICP to enhance the accuracy of estimated motion parameters. Given the estimated transformation, a global model is built and the reconstructed surface of the model is utilized by registering it with incoming depth data for next dense estimation to reduce time-evolving drift. The main contributions of this chapter are summarized as follows.

1. We propose a robust orientation estimation based on quaternion method for initial sparse estimation. We use a robust estimation to reduce the effect of noise or outliers. Using feature point detection and tracking, no prior assumption regarding the camera motion was made.

2. We introduce a novel weighted ICP method for better rate of convergence in optimization and accuracy in resulting trajectory. While the conventional ICP in [1] fails when there is no 3D features in the scene, our approach achieves robustness by emphasizing the influences of points that contain more geometric information of the scene.

3. We evaluate the proposed method using the RGB-D benchmark dataset which includes both color and depth data captured with Kinect and ground truth trajectory [33]. We show the accuracy and robustness of our proposed method and compare our results with existing methods.

The remainder of this chapter is organized as follows. Section 4.1 describes related work and addresses challenges in the pose estimation problem. Details of the proposed method are elaborated in Section 4.2. In Section 4.3, we analyze the performance of our system on an RGB-D benchmark dataset. Summary is discussed in Section 4.4.

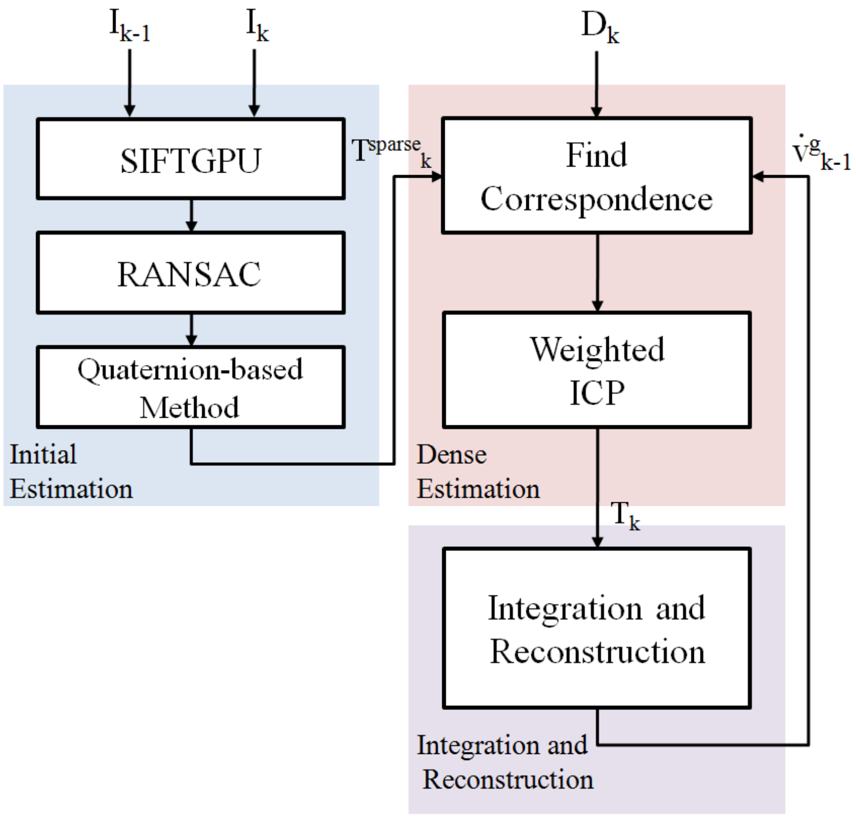


Figure 4.1: Block diagram of the proposed method.

4.1 Related Work

Recently, Kinect has become an attractive alternative to expensive range sensors. Because Kinect captures depth information in real-time and produces generally good depth map at an affordable cost many recent works on camera pose estimation have been conducted with this affordable device. We discuss specifically recent camera tracking methods using Kinect.

RGB-D Mapping system was proposed to build dense 3D model of indoor environments using Kinect [29]. The camera pose estimation is done by establishing correspondences between sparse feature points from two consecutive color frames and applying ICP to improve accuracy. Also, a view-based loop closure detection method is presented to achieve globally consistent maps. However, this method has not achieved real-time computation because of high computational complexity in the global optimization step. Therefore, it lacks the ability for user interaction and feedback.

KinectFusion [1] proposed a real-time indoor scene construction exploiting GPU programming with commodity graphics hardware. In their method, the camera pose is computed by coarse-to-fine iterative closest point algorithm (ICP), and all of range data are fused into a global volume model. They achieve less drift error without global optimization by aligning incoming depth data to the model. However, the method fails in cases where there are no significant 3D features since only depth information alone is exploited to estimate rigid body motion. Also, it fails on situation that frames are captured from remotely located cameras since small motion is assumed in the optimization.

Several different approaches have been introduced to utilize color information as well as depth data for camera pose estimation. Steinbruecker et al. [27] and Tykkala et al. [28] propose direct methods utilizing image warping function. In their methods, relative camera pose is estimated by minimizing the photometrical error between two frames.

Recently, fast visual odometry and mapping method is introduced by [30]. The system utilizes sparse feature points rather than dense data to estimate camera trajectory. The sparse features in the current frame is registered against a model of previous features and the model is updated using a probabilistic Kalman Filter. They perform the visual odometry estimation in real time (30Hz) with no GPU acceleration.

Most similar to our work is the approaches in [31, 32]. In their approach, sparse feature points of color images are extracted and matched by RANSAC. Then, the initial estimate is improved with a variant of the ICP algorithm and the resulting pose is optimized using by g^2o solver.

In most of the above approaches, the pose estimation problem is solved based on an assumption that camera motion is small or the camera moves with constant velocity. Therefore, their optimization may not converge to the optimal solution when displacement of camera between two frames become larger [1, 27].

4.2 Proposed Method

This section provides a detailed description of our approach as shown in Figure 4.1. Our camera pose estimation method mainly consists of three steps as follows.

- *Initial Sparse Estimation*: A coarse estimation of relative rigid body motion between two consecutive frames is performed. Visual feature points between two consecutive frames are detected and matched using Scale-Invariant Feature Transform (SIFT) and a relative pose is recovered using quaternion-based orientation estimation method of Faugeras and Hebert [34] and robust estimation [35].

- *Dense Estimation*: The initial estimation of camera pose is refined by the proposed weighted multi-scale ICP method. In contrast to the conventional ICP [1], we adaptively assign weights for each point to improve the rate of convergence in optimization and avoid possible failure in situations where 3D features are absent.

- *Integration and Surface Reconstruction*: Given the estimated camera pose, depth data is integrated into a single global volume using truncated distance function (TSDF) representation [36, 1]. Then, a surface geometry is reconstructed from the volume by raycast algorithm. The reconstructed surface data is used for dense estimation in the next frame to reduce time-evolving drift problem.

4.2.1 Notation and Background

We denote I_k, $k = 1, ..., K$ and D_k, $k = 1, ..., K$ as the color images and depth maps of the k-th frame, respectively, where K is the total number of the frames. The value of color image and depth map at pixel $\mathbf{p} : [u \;\; v]^\top$ in k-th frame are represented as $I_k(\mathbf{p})$ and $D_k(\mathbf{p})$, respectively. To convert each point in a depth map into a 3D point, we assume a constant intrinsic camera calibration matrix $\mathbf{K}$. Along with the depth data D, a 2D point $\mathbf{p}$ is converted into a 3D point $\mathbf{v}$ in the camera's coordinate space as shown in Eq. (4.1)

$$\mathbf{v}_k(\mathbf{p}) = D_k(\mathbf{p})\mathbf{K}^{-1} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{4.1}$$

Also, the normal vector is calculated as

$$\begin{aligned} \mathbf{n}_k(x, y) = \mathrm{Normalize}((&\mathbf{v}_k(u+1, v) - \mathbf{v}_k(u, v)) \\ &\times(\mathbf{v}_k(u, v+1) - \mathbf{v}_k(u, v))) \end{aligned} \tag{4.2}$$

where $\mathrm{Normalize}(\cdot)$ is a function which gives the normalized form of the input vector. We also denote the six-degrees-of-freedom (6DOF) camera orientation matrix at time k as $\mathbf{T}_k$, which transforms the acquired point cloud in camera space into the global space. The rigid transformation matrix $\mathbf{T}_k$ contains a rotation matrix $\mathbf{R}_k$ and a translation vector $\mathbf{t}_k$, i.e. $\mathbf{T}_k = [\mathbf{R}_k|\mathbf{t}_k]$. Finally, the 3D points and normal vector in the global coordinate system can be easily computed by the transformation matrix $\mathbf{T}_k$ as $\mathbf{v}_k^g = \mathbf{T}_k\mathbf{v}_k$ and $\mathbf{n}_k^g = \mathbf{R}_k\mathbf{n}_k$, respectively.

4.2.2 Initial Sparse Estimation

Initial pose estimation is performed using sparse feature-point detection and matching processes. Since the feature-point matching method does not require any camera motion priors to find correspondences, the points with larger displacement caused by the rapid motion (shaky frames) can be matched. To speed up detection and matching process, a SIFT method implemented with GPU programming is utilized [37]. First, SIFT feature points from two consecutive RGB images are detected and matched. We denote the corresponding feature points as $\mathbf{p}_{k-1}(i)$ and $\mathbf{p}_k(i)$, $i = 1, ..., N$, where N

is the total number of the corresponding points. Next, RANSAC is applied to find an initial transformation parameters. Using the camera matrix $\mathbf{K}$, the 2D corresponding points can be converted into 3D corresponding points. We repetitively select three random pairs of corresponding points at a time and estimate a transformation matrix [34]. We find the best initial transformation parameters which have the most inlier feature points. Given the set of inlier 3D corresponding points $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$, next step is to seek relative transformation matrix $\mathbf{T}_k^{sparse} = [\mathbf{R}_k^{sparse}|\mathbf{t}_k^{sparse}]$, minimizing the error.

$$E_k^{sparse} = \sum_i \rho(r_k(i)/\sigma_{MAD})r_k(i) \tag{4.3}$$

and

$$r_k(i) = \|\mathbf{v}_{k-1}(i) - \mathbf{R}_k^{sparse}\mathbf{v}_k(i) - \mathbf{t}_k^{sparse}\|^2 \tag{4.4}$$

M-estimator weight function ρ is employed to suppress the effect of inaccurate localization of feature points. The $\rho(\cdot)$ is the Tukey bi-weight function M-estimator, defined as

$$\rho(c) = \begin{cases} (1 - \frac{c^2}{\beta^2})^2, & \text{if } |c| \leq \beta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

where β is set to 4.6851 to obtain 95% asymptotic efficiency assuming Gaussian residual error. Since the residual errors in Eq. (4.4) does not follow a standard normal distribution, the residual errors are scaled by σ_{MAD} which is the Median Absolute Deviation (MAD) [38]. Eq. (4.3) is minimized using quaternion-based orientation method which does not require any priors or small movement assumption to estimate a rigid transformation between frames [34]. More specifically, we update $\mathbf{R}_k^{sparse}$ and $\mathbf{t}_k^{sparse}$ as follows.

1. Compute weight using Eq. (4.5)
2. Compute $\mathbf{R}_k^{sparse}$ and $\mathbf{t}_k^{sparse}$ that minimize Eq. (4.3) using [34].
3. Iterate the above steps until the update of both $\mathbf{R}_k^{sparse}$ and $\mathbf{t}_k^{sparse}$ are small.

It is important to note that inaccurate sparse points are downweighted by using robust estimation [35].

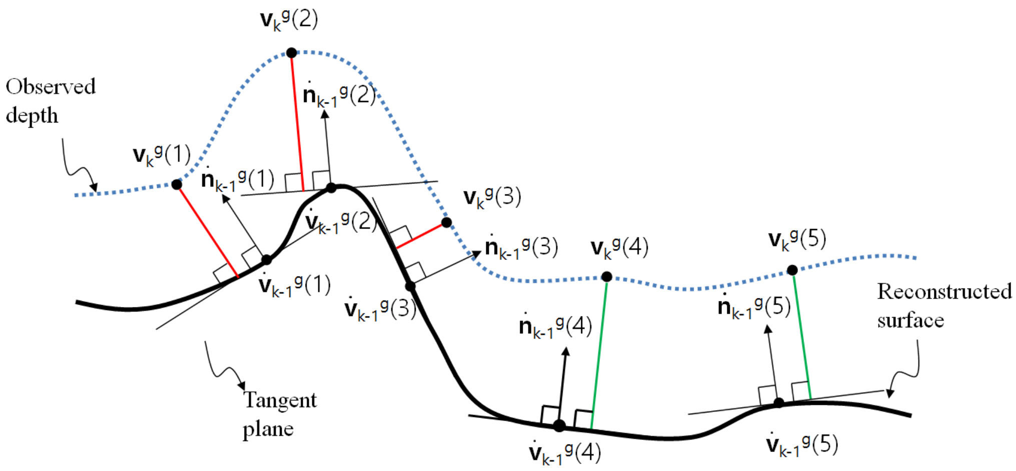

$\mathbf{v}_k{}^g(2)$
Observed depth
$\mathbf{v}_k{}^g(1)$
$\dot{\mathbf{n}}_{k-1}{}^g(2)$
$\dot{\mathbf{n}}_{k-1}{}^g(1)$
$\dot{\mathbf{v}}_{k-1}{}^g(2)$
$\mathbf{v}_k{}^g(3)$
$\dot{\mathbf{n}}_{k-1}{}^g(3)$
$\mathbf{v}_k{}^g(4)$
$\mathbf{v}_k{}^g(5)$
$\dot{\mathbf{v}}_{k-1}{}^g(1)$
$\dot{\mathbf{v}}_{k-1}{}^g(3)$
$\dot{\mathbf{n}}_{k-1}{}^g(4)$
$\dot{\mathbf{n}}_{k-1}{}^g(5)$
Reconstructed surface
Tangent plane
$\dot{\mathbf{v}}_{k-1}{}^g(4)$
$\dot{\mathbf{v}}_{k-1}{}^g(5)$

Figure 4.2: Weighted point-to-plane error between the observed depth data and the reconstructed surface data. In this example, the points at $\mathbf{p} = 1, 2, 3$ have more weights than the points at $\mathbf{p} = 4, 5$ since their geometric features are more significant.

4.2.3 Dense Estimation

Once the initial transformation has been estimated using sparse corresponding points, the transformation is refined to enhance the accuracy of the estimated motion parameters. Our dense estimation is based on the variant of ICP described in [1]. First, whole depth data is converted to point cloud using a constant camera Matrix. Then, the point-to-point correspondence between frames is recovered using projection-based matching [4]. After establishing corresponding pairs in the overlapped area, the camera pose is estimated using multi-scale ICP method. Given two point clouds, the ICP algorithm iteratively computes a rigid transformation that minimizes the mean squared Euclidean distance between corresponding points. However, the traditional ICP may converge to a local minimum for scene lacking 3D features, e.g., a large flat area such as a wall or desk surface. Since the method assumes that all points are equally important in the ICP method, it would lead to misalignment in the result. If the majority of corresponding points are selected from featureless regions and used for pose estimation, the algorithm converges slowly or finds inaccurate pose because of lack of constrains.

Instead of selecting a subset of points, we take into account the significance of corresponding points in terms of the contribution to the residual errors to avoid local minimum. In our approach, a weight is adaptively assigned to each point based on the criteria that points around the edges of objects often have high geometric features. The weight for a point $\mathbf{p}$ is determined by the following equation.

$$\xi_k(\mathbf{p}) = 1 - \sum_{\substack{\mathbf{q} \notin \Phi, \\ \mathbf{q} \in \Omega_{\mathbf{p}}}} \Pi(|D_k(\mathbf{p}) - D_k(\mathbf{q})|, \sigma_D(\mathbf{p}))/(\text{\# of } \mathbf{q}) \tag{4.6}$$

where Φ is uncertain region where Kinect cannot capture depth information and $\Omega_{\mathbf{p}}$ is the window centered at pixel $\mathbf{p}$ with an empirically chosen size. Π is the Gaussian function with variance $\sigma_D(\mathbf{p})$. The variance is chosen adaptively based on an observation that the random error of depth measurements increases quadratically with increasing distance from Kinect camera [39].

$$\sigma_D(\mathbf{p}) = \alpha D_k(\mathbf{p})^2 \tag{4.7}$$

where α is 2.85×10^{-5} when the unit of D is millimeters.

Finally, a rigid transformation matrix $\mathbf{T}_k^{dense}$ is obtained by minimizing the point-to-plane error metric [40] with the weight, which is defined as the weighted sum of squared errors between each point in the current data and the tangent plane at its corresponding point in the surface model reconstructed from previous frame (Figure 4.2).

$$E_k^{dense} = \sum_{\mathbf{p}} \xi_k(\mathbf{p}) \|(\dot{\mathbf{v}}_{k-1}^g(\mathbf{p}) - \mathbf{T}_k^{dense} \mathbf{v}_k^g(\mathbf{p})) \cdot \dot{\mathbf{n}}_{k-1}^g(\mathbf{p})\|^2 \tag{4.8}$$

where $\dot{\mathbf{v}}_{k-1}^g$ and $\dot{\mathbf{n}}_{k-1}^g$ denote corresponding 3D points and their normal vectors in the surface model, respectively. Since we have coarsely aligned two points clouds using initial estimation, we can safely assume that the displacement between frames is very small. The minimization problem of Eq. (5.10) can be approximated by linearizing it [1].

Algorithm 1 summarizes the overall dense estimation process. For each scale, the current camera transformation matrix $\mathbf{T}_k$ is estimated. The current camera pose is updated with incremental transform $\mathbf{T}_k^{dense}$ per iteration by registering a newly transformed point cloud to the previous frame based on the geometry in overlapping areas. The resulting transformation matrix from coarser scale is used as an initial guess for finer scale. The maximum numbers of iteration for level 2, 1, and 0 are 8, 10, and 20, respectively.

Algorithm 1 Dense Estimation from current RGB-D image and the surface model

```
input: $\mathbf{T}_k^{sparse}, \mathbf{T}_{k-1}, \dot{\mathbf{v}}_{k-1}^g, \dot{\mathbf{n}}_{k-1}^g, \mathbf{v}_k, \mathbf{n}_k$
$\mathbf{T}_k \leftarrow \mathbf{T}_k^{sparse}\mathbf{T}_{k-1}$
for $l$ = 2 (coarsest level) to 0 (finest level) do
    Decimate $\mathbf{v}_k, \mathbf{n}_k, \dot{\mathbf{v}}_{k-1}^g, \dot{\mathbf{n}}_{k-1}^g$ by decimation factor $2^l$
    for $iteration$ = 1 to $max\_iteration$ do
        $\mathbf{v}_k^g \leftarrow \mathbf{T}_k\mathbf{v}_k$
        $\mathbf{n}_k^g \leftarrow \mathbf{R}_k\mathbf{n}_k$
        Find point correspondences using [4]
        Compute weight $\xi_k$ using Eq. (5.8)
        Estimate transformation $\mathbf{T}_k^{dense}$ minimizing Eq. (5.10)
        $\mathbf{T}_k \leftarrow \mathbf{T}_k^{dense}\mathbf{T}_k$
    end for
end for
```

4.2.4 Integration and Surface Reconstruction

In most real-time odometry methods, the camera pose estimation is performed on a frame-by-frame basis. Therefore, motion error is accumulated over time (e.g., time-evolving drift), since the camera path is computed by chaining the relative pose between consecutive frames.

Given the estimated global camera pose in the previous stage, the geometric information of the current scene is transformed into a single volume space, consisting of small cubical volume elements called voxels. Then, the transformed information is projected into each corresponding voxel in a global volume with a value determined by volumetric truncated signed distance function (TSDF) [1]. TSDF represents the truncated signed distance of each voxel to the nearest surface along the line between the voxel and camera origin.

For dense surface reconstruction, raycasting method is used [1] in our approach. For every pixel of the image, a ray is sent to the global volume with a direction from the camera origin to the pixel coordinate of the image. When it intersects a surface on its way, the surface information is obtained from the global volume and the reconstructed surface is used for surface reconstruction.

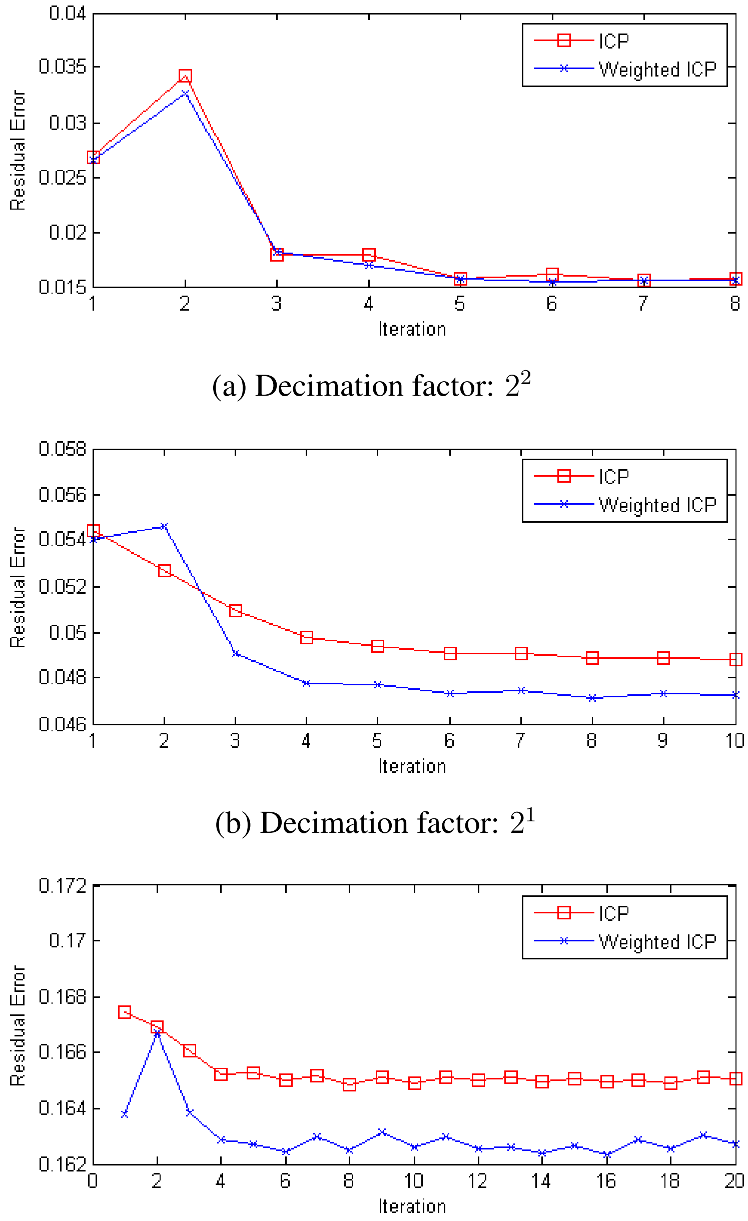
4.3 Experiments

In this section, we describe the experiments performed to evaluate the proposed method. We first demonstrate the benefits of the proposed weighting function. Then, we compare the performance of our method with state-of-art methods. Finally, runtime evaluation of the proposed method is presented. In our experiments, we use an RGB-D benchmark dataset [33] which includes calibrated and registered RGB-D sequences captured from Kinect and ground truth trajectory data obtained from a high-accuracy motion-capture system.

4.3.1 Conventional ICP and Weighted ICP

To demonstrate the benefits of our weighting function, we evaluate the performance for *freiburg1_desk* sequence which contains scenes with fewer number of geometric features. Figure 4.3 shows the convergence profile of the conventional ICP [1] and our weighted ICP method for the dataset. In this test case, each iteration's residual error for whole frames is averaged. The y-axis is the averaged residual error for each iteration, and the x-axis is iteration number. As shown in the figure, the proper weight notably improves the rate of convergence.

To demonstrate the accuracy improvement, we evaluate relative pose error (RPE) to compare drift errors between traditional ICP and the weighted ICP on *freiburg1_desk* dataset. Figure 4.4(a) and (b) shows a RGB-D sequence with fewer number of geometric features (Frames 240 to 290 of *freiburg1_desk* dataset) and the relative pose error comparison in sequences is shown in Figure 4.4(c). As seen in this figure, the relative pose error is reduced in the most of feature-less frames by using the proposed weighting function.

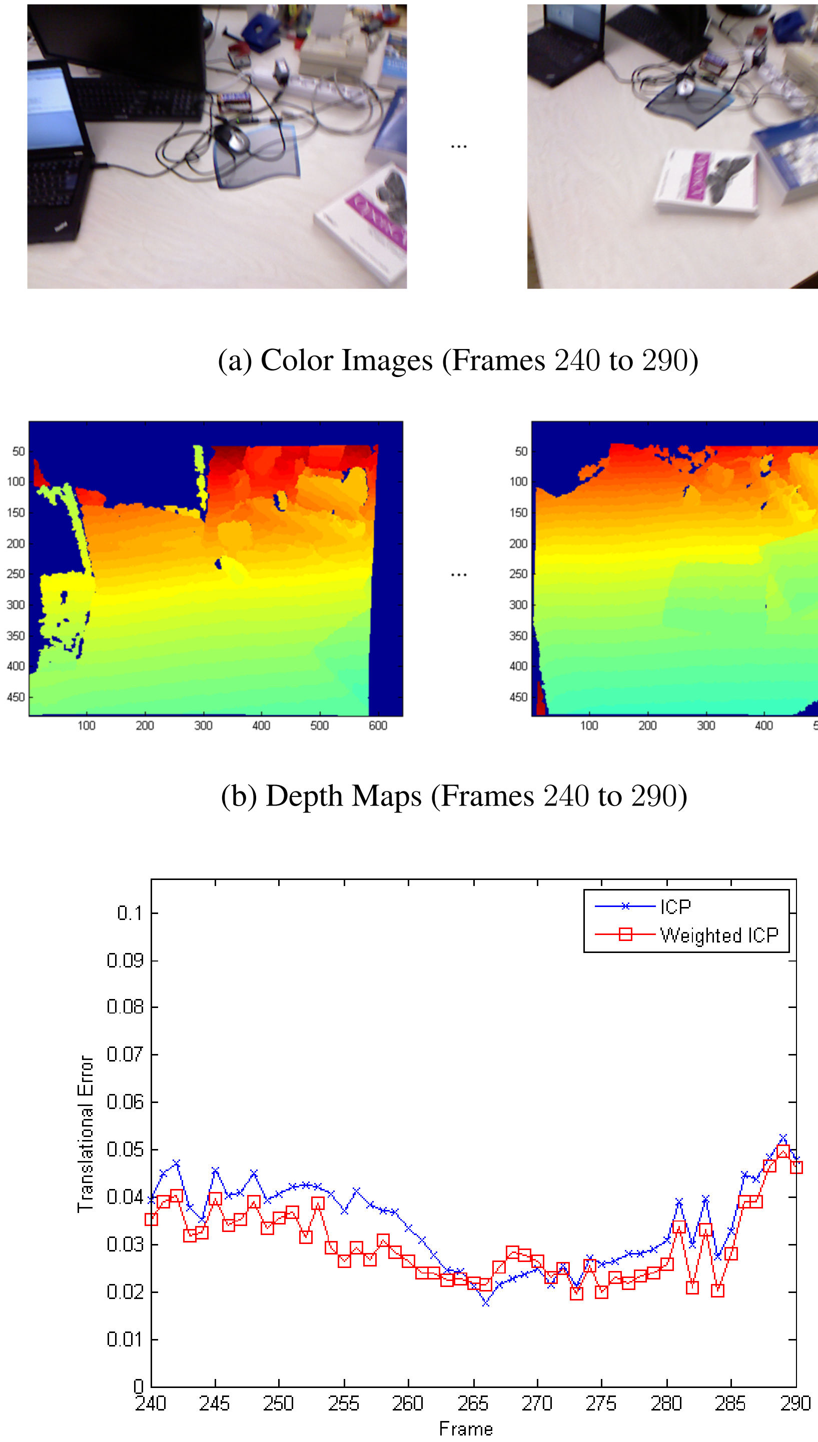


(a) Decimation factor: 2^2

(b) Decimation factor: 2^1

(c) Decimation factor: 2^0

Figure 4.3: Convergence profile of the conventional ICP [1] and our weighted ICP method for *freiburg1_desk*. Note that the proposed method improves the rate of convergence.



(a) Color Images (Frames 240 to 290)

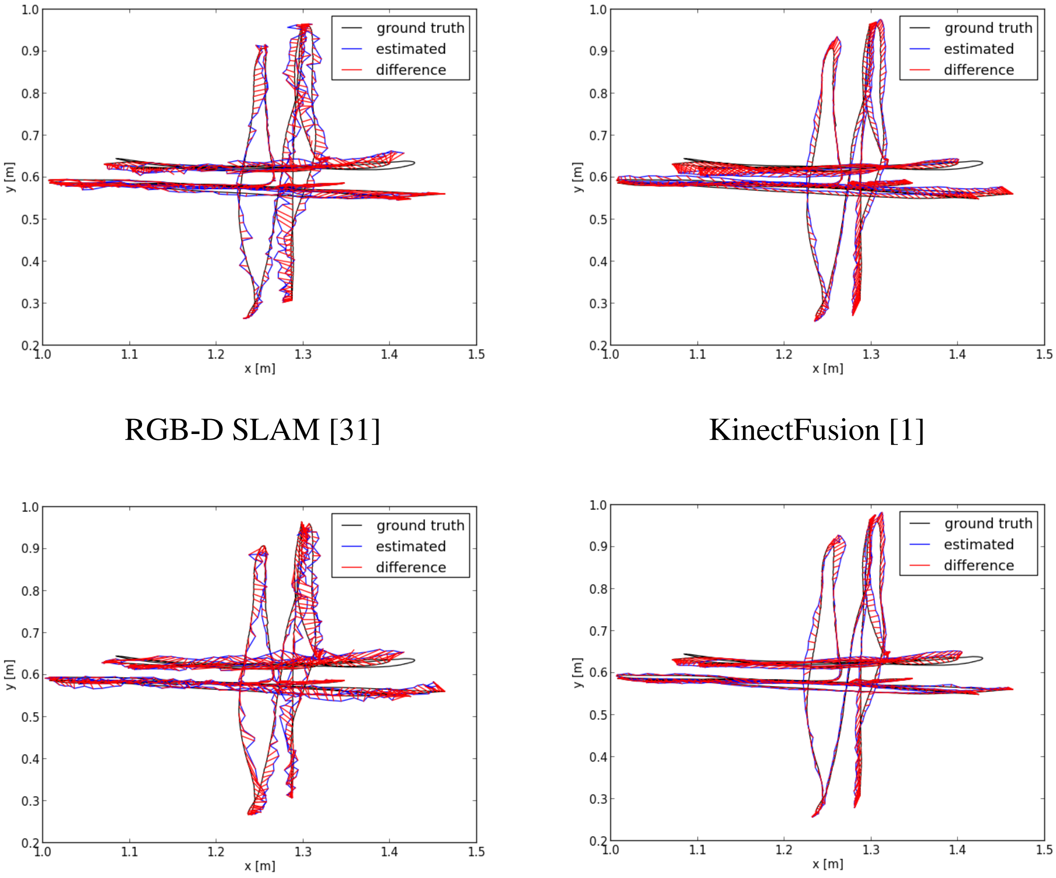
(b) Depth Maps (Frames 240 to 290)

(c) Relative translational pose error

Figure 4.4: Relative pose error comparison between traditional ICP and our weighted ICP on scenes with fewer number of geometric features of *freiburg1_desk* dataset (Frames 240 to 290).

Table 4.1: Comparison results of our approach with other methods in absolute trajectory error (ATE).

		freiburg1_xyz	*freiburg1_desk*	*freiburg1_plant*
RMSE (mm)	RGB-D SLAM [31]	13.473	25.831	61.407
	KinectFusion [1]	15.612	224.194	187.545
	CCNY RGBD [30]	17.339	106.186	93.893
	Proposed Method	13.346	20.419	41.043
Mean (mm)	RGB-D SLAM [31]	12.029	23.132	55.567
	KinectFusion [1]	14.005	44.376	163.632
	CCNY RGBD [30]	15.302	92.005	83.861
	Proposed Method	11.75	18.661	37.127
Median (mm)	RGB-D SLAM [31]	11.176	21.388	51.533
	KinectFusion [1]	12.432	23.440	152.564
	CCNY RGBD [30]	14.004	73.002	75.237
	Proposed Method	10.341	17.889	35.296
Std. Dev. (mm)	RGB-D SLAM [31]	6.068	11.497	26.135
	KinectFusion [1]	6.901	219.758	91.639
	CCNY RGBD [30]	8.153	53.014	42.228
	Proposed Method	6.330	8.289	17.803



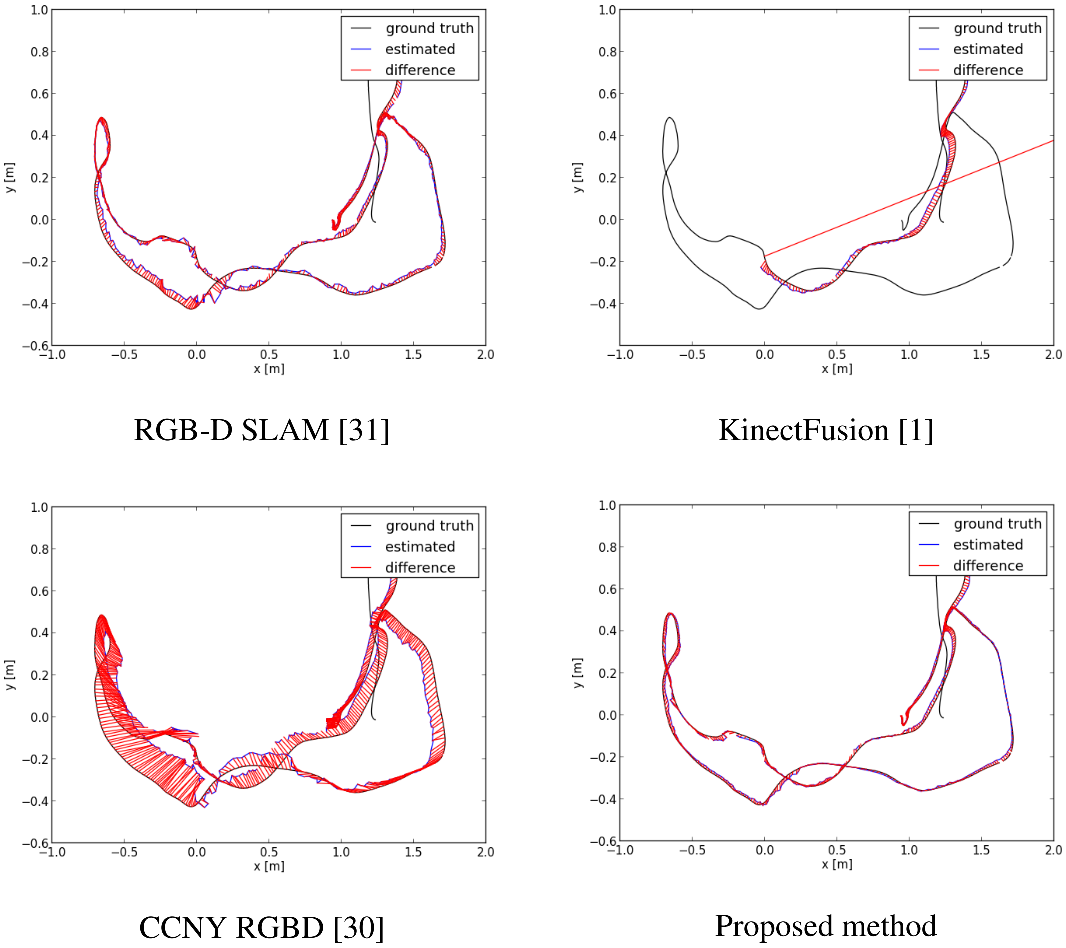
RGB-D SLAM [31]

KinectFusion [1]

CCNY RGBD [30]

Proposed method

Figure 4.5: Ground truth and estimated camera trajectories of *freiburg1_xyz*.



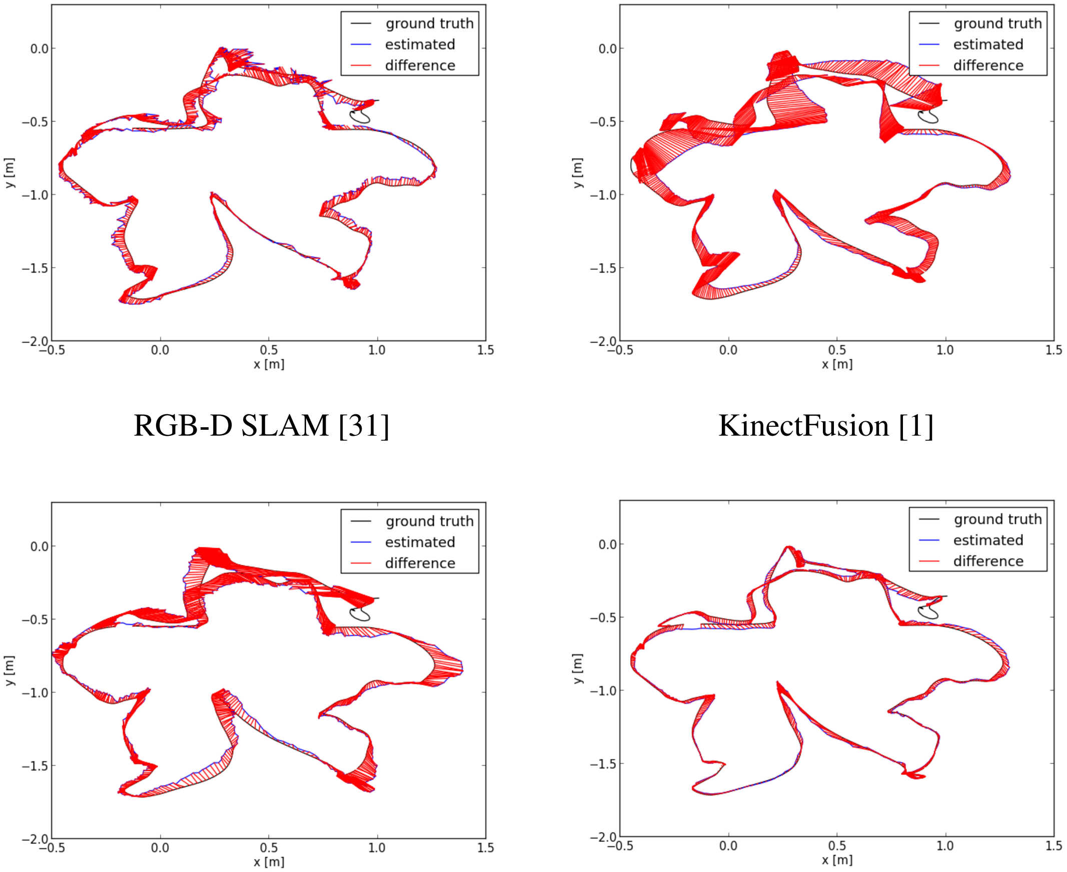
RGB-D SLAM [31]

KinectFusion [1]

CCNY RGBD [30]

Proposed method

Figure 4.6: Ground truth and estimated camera trajectories of *freiburg1_desk*.



RGB-D SLAM [31]

KinectFusion [1]

CCNY RGBD [30]

Proposed method

Figure 4.7: Ground truth and estimated camera trajectories of *freiburg1 _plant*.

4.3.2 Quantitative Comparisons

For a quantitative comparison, we compare the performance of our approach with three different methods that are RGB-D SLAM [31], KinectFusion [1], and CCNY RGBD [30]. Since original KinectFusion cannot handle large scale area, an extended KinectFusion with capability of shifting the global volume based on an implementation in the Point Cloud Library (PCL) [1] is used.

We select three sequences from the benchmark dataset that are *freiburg1_xyz*, *freiburg1_desk*, and *freiburg1_plant* because they are real world examples with various scenes and fast camera motion. Average translation velocities for *freiburg1_xyz*, *freiburg1_desk*, and *freiburg1_plant* are $0.24m/s$, $0.41m/s$, and $0.37m/s$, respectively.

Figures 4.5, 4.6, and 4.7 show the ground truth and estimated trajectories projected onto the x-y plane. The ground truth trajectories, estimated trajectories, and the differences are presented in black, blue, and red lines, respectively. Trajectories generated from RGB-D SLAM method are inaccurate because the motion is estimated on a frame-by-frame basis. KinectFusion method fails to track camera motion when the camera is moving fast such as in dataset *freiburg1_desk* and *freiburg1_plant*. CCNY RGBD method is also very inaccurate since the method uses only sparse points in their estimation rather than dense data. On the other hand, the proposed method yields smoother and more accurate trajectory due to the accumulated global volume data and is more robust to fast motion because of sparse feature-based estimation.

Table 4.1 shows the absolute trajectory error (ATE) measuring the difference between points of the ground truth and the estimated trajectory. This table verifies that our approach achieves better accuracy than other approaches. Note that the trajectory of RGB-D SLAM is unstable in a plot with a finer scale even though our RMSE improvement over RGB-D SLAM method on *freiburg1_xyz* is small.

4.3.3 Computational Performance

We use a computer with Intel Core i7 2.93GHz processor with 12GB RAM and NVidia GTX 680 graphic card with 2GB GPU memory in all experiments. By im-

[1] http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php

plementing our approach with GPU programming, we have achieved a near real-time performance. Our system runs on average at 12.78Hz. Average timing for each process are: SIFTGPU: $25.45ms$, RANSAC: $3.17ms$, Quaternion-based method: $3.53ms$, ICP: $36.32ms$, and Integration: $9.81ms$.

4.4 Summary

In this chapter, we propose a robust method for camera tracking and surface mapping method to cope with challenging situations such as fast camera motion or geometrically featureless scenes. In our approach, sparse feature-based estimation is used for initial pose estimation which handle frames that are captured with a fast camera motion. Then, we improve the accuracy of the estimated transformation matrix by using dense data and assigning weights adaptively according to the importance of each point. Finally, a global surface model is built with the estimated transformation and reconstructed surface model is used for next dense estimation for less drift errors. Experimental results show that our method outperforms state-of-the-art approaches significantly.

4.5 Acknowledgments

This chapter is in part a reprint of the following publication: Kyoung-Rok Lee, Truong Q. Nguyen, “Robust Tracking and Mapping with a Handheld RGB-D Camera”, *IEEE Winter Conference on Applications of Computer Vision*, Steamboat Spring, CO, Mar. 2014.

Chapter 5

Reconstruction of Static Objects

The 3D reconstruction of real objects or scenes is a fundamental problem in the field of computer vision and graphics. Using an image sequence captured from arbitrary view points, the goal is to build geometry and appearance of 3D model.

Recently, the KinectFusion method has been proposed for real-time scene reconstruction by exploiting GPU programming with common graphics hardware [1]. In the method, the camera pose is estimated by a coarse-to-fine iterative closest point algorithm (ICP), then all the range data is integrated into a single global volume using volumetric representation. The resulting surface model is reconstructed from the volume by ray-cast algorithm. One drawback of KinectFusion is that such methods suffer from a lack of realism with low quality depth maps, since the Kinect is only able to acquire surface depths from areas on which the infrared (IR) speckle points have been sampled. These limitations reduce the quality of surface geometry by discarding desired complex and finer-scale textures. Furthermore, the camera tracking method of the KinectFusion may fail when a scene has very few 3D features. In other words, if the majority of points are located in featureless regions such as a wall or desk surface, the algorithm may fail to find an optimal solution, because of its lack of constraints.

We propose a novel approach for realistic surface geometry reconstruction using RGB-D images. While the proposed method follows the framework of KinectFusion, it differs in three ways:

1. We utilize high quality RGB images by attaching an HD RGB camera onto a

Kinect to reconstruct a 3D model with realistic surface geometry and color textures.

2. We extend the depth map refinement presented in Chapter 3 and exploit a robust camera tracking method presented Chapter 4 to achieve accurate camera pose estimations.

3. We evaluate the proposed method using our real object dataset, which has been generated by laser scans.

The rest of this chapter is organized as follows. Section 5.1 discusses previous methods in 3D reconstruction of static scene. Section 5.2 briefly describes an overview of the proposed system. In Section 5.3, we describe the proposed surface geometry refinement method to recover finer-scale surface geometry. In Section 5.4, we describe a robust camera tracking method that can find accurate camera poses. In Section 5.5, we describe integration method to fuse all live data into a global volume space and reconstruct surface models from the global volume. In Section 5.6, we demonstrate our experimental results. Finally, summary are presented in Section 5.7.

5.1 Related Work

Recently, affordable depth sensing devices such as Time-of-Flight (ToF) [8] and Microsofts Kinect [7] have been rapidly adopted by computer vision researchers. These low-price easy-to-use range sensing technologies enables many system capabilities for both customers and researchers. Cui et al. presented a 3D object scanning system using a ToF camera [3]. In their system, data captured by moving the camera around an object is integrated into a single model through a probabilistic scan alignment approach. In addition, a resolution enhancement method using sensor's noise characteristic to achieve 3D model with reasonable quality, is proposed.

A scanning system for capturing 3D full human body models using multiple Kinects is proposed in [41]. They constructed a rough mesh template and updated the model by deforming it in successive frames. Global alignment was exploited to solve the loop closure problem by distributing errors in the deformation space.

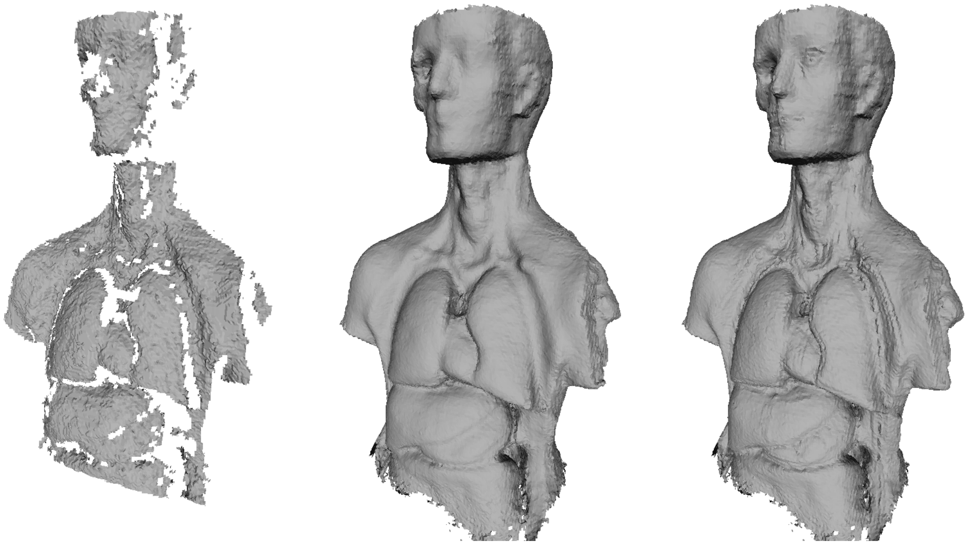


Figure 5.1: Left: The reconstructed surface model from the raw data of capturing anatomy with Kinect camera. Middle: The reconstructed surface model anatomy generated using KinectFusion. Right: The improved surface model generated using our method.

Even though the above approaches created high-quality models from difference scan views, they did not achieve real-time computation, because of the high computational complexity. Therefore, their methods lack the possibility for user interaction and feedback.

In [42], a 3D capturing system using a tracked Kinect is proposed. They achieved real-time camera tracking with an AR tracking system[1]. A tracker is attached to the top of Kinect, and four ceiling-mounted infrared cameras are used to estimate the Kinects precise camera pose. Then, triangle meshes with texture maps are generated by integrating scans into a global point cloud and connecting nearby points. However, this real-time tracking method requires complex environmental setups, which restricts the capability of common usage in daily life.

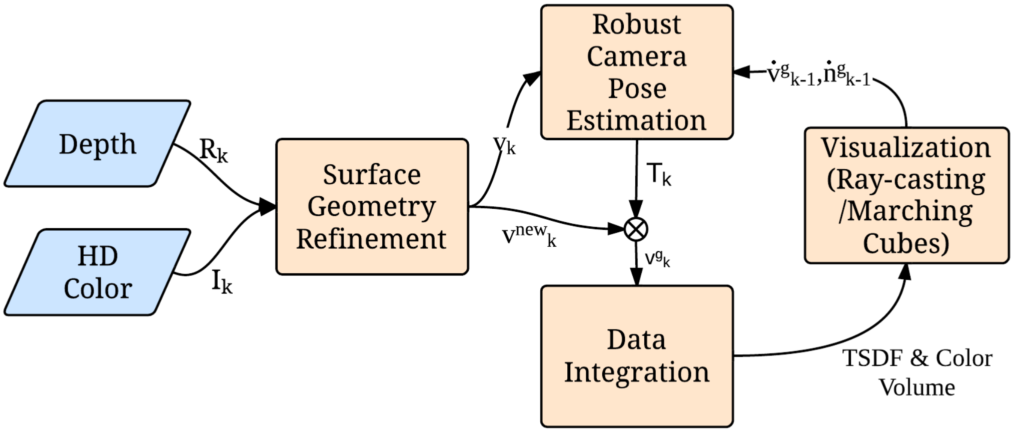

Depth
HD Color
R_k
I_k
Surface Geometry Refinement
v_k
v^{new}_k
Robust Camera Pose Estimation
T_k
v^g_k
$\dot{v}^g_{k-1}, \dot{n}^g_{k-1}$
Visualization (Ray-casting /Marching Cubes)
Data Integration
TSDF & Color Volume

Figure 5.2: Flow chart.

5.2 System Overview

The Kinect sensor consists of an RGB camera with a 1.3 megapixel (MP) resolution, an infrared (IR) camera with resolution of 1.3 MP, and an IR pattern projector. Figure 5.1(a) shows the reconstructed surface model from the raw depth data from Kinect. To obtain high-resolution RGB images, an additional HD camera with a 5 MP resolution is attached to the top of a Kinect and is calibrated with the Kinect IR camera, using a checkerboard pattern [43]. Figure 5.3 shows the components of our Kinect device with an HD RGB camera.

Figure 5.2 shows an overview of the proposed 3D reconstruction system. Our reconstruction method mainly consists of four stages, which are surface geometry refinement, robust camera tracking, data integration, and visualization.

- Surface Geometry Refinement: The live range data is captured by either handheld scanning or scanning under controlled motion (with a turntable). The raw depth map is refined by utilizing high frequency information in color images.

- Robust Camera Tracking: Using a weighted ICP method, the orientation of the camera in the global coordinate system is estimated for registration between frames. Unlike conventional ICP, the weight is assigned to each point to achieve robust-

[1] http://www.ar-tracking.com/home/

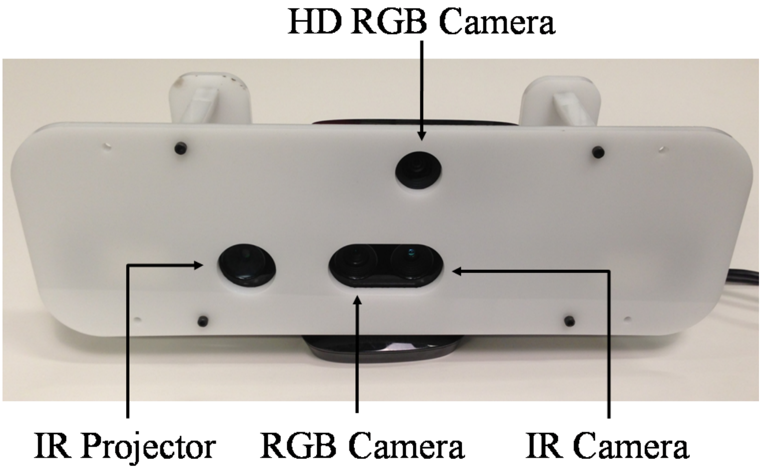


Figure 5.3: Components of Kinect with an HD camera. The top HD RGB camera is used to capture high quality color images to refine surface geometry and integrate color texture.

ness by emphasizing the influence of points that contain more of the scenes geometric information.

- Data Integration: The aligned data is fused together to update a single global 3D volume.
- Visualization: The global volume is visualized as either a rendered image or a mesh model.

5.3 Surface Geometry Refinement

Since the Kinect only sparsely samples the scene using its IR speckle pattern, there is the possibility that finer object details are missing, and data near an object's boundaries may not be accurate. This problem degrades the overall surface geometry reconstruction results. On the other hand, color images from HD RGB cameras can present much finer-scale information.

In Chapter 4, we proposed the sampling-based robust multi-lateral filter (SRMF), which is a method of refining depth maps using high-resolution color images. In the method, unreliable regions are first defined by calculating the reliability measure for each pixel in the depth map. Then, samples from the region of high reliability are collected, and the best sample with the highest fidelity is selected. Finally, a robust multi-lateral filter, which is an extended joint bilateral filtering technique, is applied to reduce noise while preserving the edges' sharpness. Although SRMF successfully improved the quality of depth maps, the method is primarily focused on the refinement of objects boundaries. The color image's accurate boundaries are used as guidance to fix errors in the depth maps transition areas. Instead, one can utilize high frequency information in color images to estimate an object's geometry.

Our work is inspired by Beeler et al. [44]. In their method, surface geometry is refined based on the observation that the intensities of images in small concavities tend to be darker when we assume albedo is uniform in the surface, since the light reflected by a diffuse surface is partially obstructed. Our surface geometry refinement method takes a color image I and a depth data R as inputs. First, SRMF is applied to raw depth to reduce noise in an object's boundaries while recovering precise edges. We denote the filtered depth map as $D(\mathbf{p})$. To convert each pixel $\mathbf{p} : [\, u \;\; v \,]^\top$ in the filtered depth data into a 3D point $\mathbf{v}(\mathbf{p})$, we assume a constant intrinsic camera calibration matrix $\mathbf{K}$, which transforms 3D points in the camera coordinate system into the image plane. Given the intrinsic camera calibration matrix $\mathbf{K}$, a back-projected 3D point $\mathbf{v}(\mathbf{p})$ in the camera space is computed as:

$$\mathbf{v}(\mathbf{p}) = D(\mathbf{p})\mathbf{K}^{-1} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{5.1}$$

In addition, the normal vector at pixel $\mathbf{p}$ is calculated as:

$$\begin{aligned} \mathbf{n}(y, x) = \mathrm{Normalize}((&\mathbf{v}(u+1, v) - \mathbf{v}(u, v)) \\ &\times (\mathbf{v}(u, v+1) - \mathbf{v}(u, v))) \end{aligned} \tag{5.2}$$

where $\mathrm{Normalize}(\cdot)$ is a function that gives the normalized form of the input vector.

For geometry refinement, we first find high frequency features in color image using a high-pass filter. The filter is implemented by subtraction of a blurred version of an original image from the original image itself. The blurred image is generated by convolving the original I with a Gaussian kernel, i.e. $\mathcal{N}(\mathbf{0}, \sigma_g^2\mathbf{I})$. We denote $\mu(\mathbf{p})$ as a high frequency feature, and it is computed as:

$$\mu(\mathbf{p}) = I(\mathbf{p}) - [I * \mathcal{N}](\mathbf{p}) \tag{5.3}$$

where $*$ denotes the convolution operation in x and y spaces.

Next, based on our assumption that gradient changes in image intensity are related to variations in the geometry, we define a correction factor as:

$$\delta(\mathbf{p}) = \eta \frac{\sum_{\mathbf{q}\in\Omega_\mathbf{p}} w(\mathbf{p},\mathbf{q}) f_\mu(\mathbf{p},\mathbf{q}) f_\xi(\mathbf{p},\mathbf{q})}{\sum_{\mathbf{q}\in\Omega_\mathbf{p}} w(\mathbf{p},\mathbf{q})} \tag{5.4}$$

where $\Omega_\mathbf{p}$ is the neighborhood of $\mathbf{v}(\mathbf{p})$, η is an embossing parameter that controls the strength of the correction factor, and $w(\mathbf{p},\mathbf{q})$ is a radial basis function, defined as $e^{-\|\mathbf{v}(\mathbf{p})-\mathbf{v}(\mathbf{q})\|}$.

The weight f_μ can be computed, based on the criteria that the surface geometry forms a concavity when $\mu(X)$ decreases, and it is likely to form a convexity when $\mu(X)$ increases.

$$f_\mu(\mathbf{p},\mathbf{q}) = \mu(\mathbf{p}) - \mu(\mathbf{q}) \tag{5.5}$$

The weight f_ξ is used to attenuate Eq. (5.5) when the geometric gradient is significant.

$$f_\xi(\mathbf{p},\mathbf{q}) = 1 - \frac{| < \mathbf{v}(\mathbf{p}) - \mathbf{v}(\mathbf{q}), \mathbf{n}(\mathbf{p}) > |}{\|\mathbf{v}(\mathbf{p}) - \mathbf{v}(\mathbf{q})\|} \tag{5.6}$$

Eq. (5.6) will be close to zero when the angle between a vector $\mathbf{v}(\mathbf{p}) - \mathbf{v}(\mathbf{q})$ and $\mathbf{n}(\mathbf{p})$ is small, but the value will be one if the angle is 90°. Therefore, Eq. (5.4) will create a large response when there is little or no geometric variation.

Finally, a new 3D point is updated using the normal as:

$$\mathbf{v}^{new}(\mathbf{p}) = \mathbf{v}(\mathbf{p}) + \delta(\mathbf{p})\mathbf{n}(\mathbf{p}) \tag{5.7}$$

The refined geometry data is aligned in the global space after the camera pose estimation, and is integrated into a global 3D volume in our reconstruction method.

5.4 Camera Tracking

For a real-time, accurate 3D reconstruction method, we require fast, robust camera pose estimation method. The ICP algorithm is widely used for 3D alignment, because the algorithm is conceptually simple and fast. After the first introduction of the

ICP algorithm [45, 46], many variants have been introduced [47]. KinectFusion uses a coarse-to-fine ICP algorithm for its camera pose estimations [1]. By closely analyzing the ICP method in [1], we observe that the method fails when the scene lacks geometric features. In our system, we use a robust camera pose estimation method that uses weighted ICP, which is discussed in the previous chapter. While the conventional ICP of a KinectFusion assumes that every point has the same importance, the weighted ICP method assigns a weight to each point to improve the accuracy of the camera tracking. The weight for a point $\mathbf{p}$ in a depth map at time k is determined by:

$$\xi_k(\mathbf{p}) = 1 - \sum_{\substack{\mathbf{q}\notin\Phi,\\ \mathbf{q}\in\Omega_\mathbf{p}}} \Pi(|D_k(\mathbf{p}) - D_k(\mathbf{q})|, \sigma_D(\mathbf{p}))/(\text{\# of } \mathbf{q}) \tag{5.8}$$

where Φ is an unknown region in which the Kinect cannot capture depth value, and $\Omega_\mathbf{p}$ is the box window centered at pixel $\mathbf{p}$ with an empirically chosen size. Π is the Gaussian function with variance $\sigma_D(\mathbf{p})$ which is chosen based on the observation that the random error of depth measurements will increase quadratically as the distance from Kinect camera increases [39].

$$\sigma_D(\mathbf{p}) = \alpha D_k(\mathbf{p})^2 \tag{5.9}$$

where α is 2.85×10^{-5}; (D is in millimeters).

Finally, a global transformation matrix $\mathbf{T}_k$ is calculated by minimizing the point-to-plane error metric [40] with the weight.

$$E_k = \sum_{\mathbf{p}} \xi_k(\mathbf{p}) \|(\dot{\mathbf{v}}^g_{k-1}(\mathbf{p}) - \mathbf{T}_k\mathbf{v}_k(\mathbf{p})) \cdot \dot{\mathbf{n}}^g_{k-1}(\mathbf{p})\|^2 \tag{5.10}$$

where $\dot{\mathbf{v}}^g_{k-1}$ and $\dot{\mathbf{n}}^g_{k-1}$ represent the corresponding 3D points and their normal vectors in the generated surface model, respectively. Assuming that camera motion is slow, the minimization problem of Eq. (5.10) can be approximated and linearized [1].

5.5 Integration and Visualization

Given the estimated global camera pose, the aligned geometric information and color information require integration in a single domain. Volumetric representation is widely used for data integration in 3D reconstruction [48]; in the volumetric representation, 3D models are built in a single volume space that consists of small cubical volume elements called voxels. In our system, the Truncated Signed Distance Function (TSDF) is utilized [1]. The value of each voxel in the 3D volume represents the truncated signed distance of each voxel to the nearest surface, along the line between the voxel and camera origin. During the integration process, the data in voxels is assigned by new data, or is incrementally updated by averaging if data exists in the voxel.

The color data from the Kinect is also transformed with the estimated transformation parameters from the previous camera tracking stage, and each color datum is integrated into the corresponding voxel in a separate color volume, which has same dimensions as the TSDF volume. Our color texture integration method is based on an implementation of KinectFusion in the Point Cloud Library (PCL)[2]. For every 3D point that exists within the distance of μ to the surface, the color data is fused to the corresponding voxel in the color volume by either assigning new data or updating the existing data by averaging their values with a constant weight. In our implementation, we extend the existing color integration method by employing a spatial weight function to improve the color texture quality. The weight $\lambda(\mathbf{p})$ is likely to have a high value when the 3D point converted from $\mathbf{p}$ is spatially close to the surface of the object, and computed as

$$\lambda(\mathbf{p}) = f_c(|\text{TSDF}(\mathbf{p})|) \tag{5.11}$$

where f_c is taken to be a Gaussian function with standard deviations σ_c. The TSDF($\mathbf{p}$) represents the truncated signed distance to the surface of the objects. By accumulating the values with this weight, we can build accurate 3D representations that are consistent with all views.

5.6 Experimental Results

First, we demonstrate the benefits of the proposed surface geometry refinement method using the Middlebury dataset [9]. Then, we compare the accuracy of the proposed method with state-of-art methods, using real statue objects. Finally, we present

[2] http://www.pointclouds.org/news/kinectfusion-open-source.html

computational analysis. Our system uses an Intel Core i7 2.93GHz processor with 12GB RAM and NVidia GTX 680 graphics card with 2GB GPU memory. The depth data is acquired using the Kinect for XBOX360 and the color data is captured using Logitech HD webcam.

5.6.1 Depth Refinement Results on Middlebury Dataset

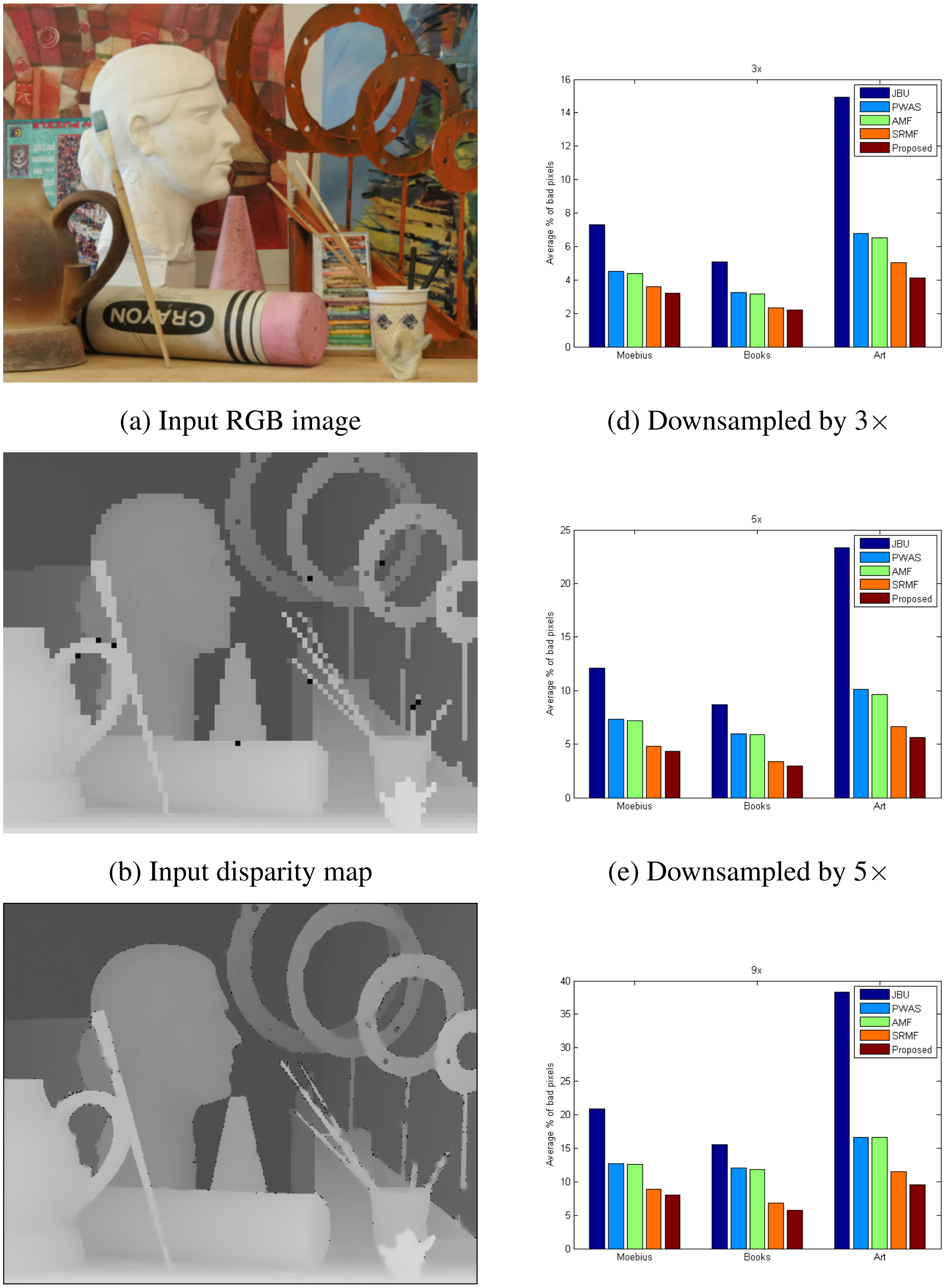
To prove the effectiveness of the proposed surface refinement method, we apply our method to the Middlebury datasets *Moebius*, *Books*, and *Art*. First, we generate downsampled versions of the provided ground truth disparity maps with a factor rate of $3\times$, $5\times$, and $9\times$, and upsample them using a nearest-neighborhood method to produce the initial estimate of the interpolated result. The original corresponding color images were directly used as high-quality guidance images. For each factor, the average percentage of bad pixels with an error threshold of 1 for all known regions is calculated. Figure 5.4 shows the evaluation results of the proposed method and other depth refinement methods, which are Joint Bilateral Upsampling (JBU) [11], Pixel Weighted Average Strategy (PWAS) [17], and Adaptive Multi-lateral Filtering (AMF) [19]. The comparison plots show that the proposed method outperforms the other refinement methods.

5.6.2 3D Reconstruction Results

We compare the resulting mesh models from the proposed approach with two different methods, which are open-source KinectFusion from PCL and SCENECT[3] from FARO. Since the trial version of SCENECT only produces registered point clouds from RGB-D video input, we utilize a screened Poisson method [49] to create watertight surface mesh models from the point cloud for fair comparison.

For the reconstruction, the Kinect and the HD RGB camera capture different statue models rotating counterclockwise on a turntable. Final mesh models generated from different 3D reconstruction methods are shown in Figure 5.5. As presented in the figure, our reconstruction system produces 3D models of better quality comparing to the other methods. Models from both KinectFusion and SCENECT have distortion due

[3] http://www.faro.com/scenect/scenect



(a) Input RGB image

(d) Downsampled by 3×

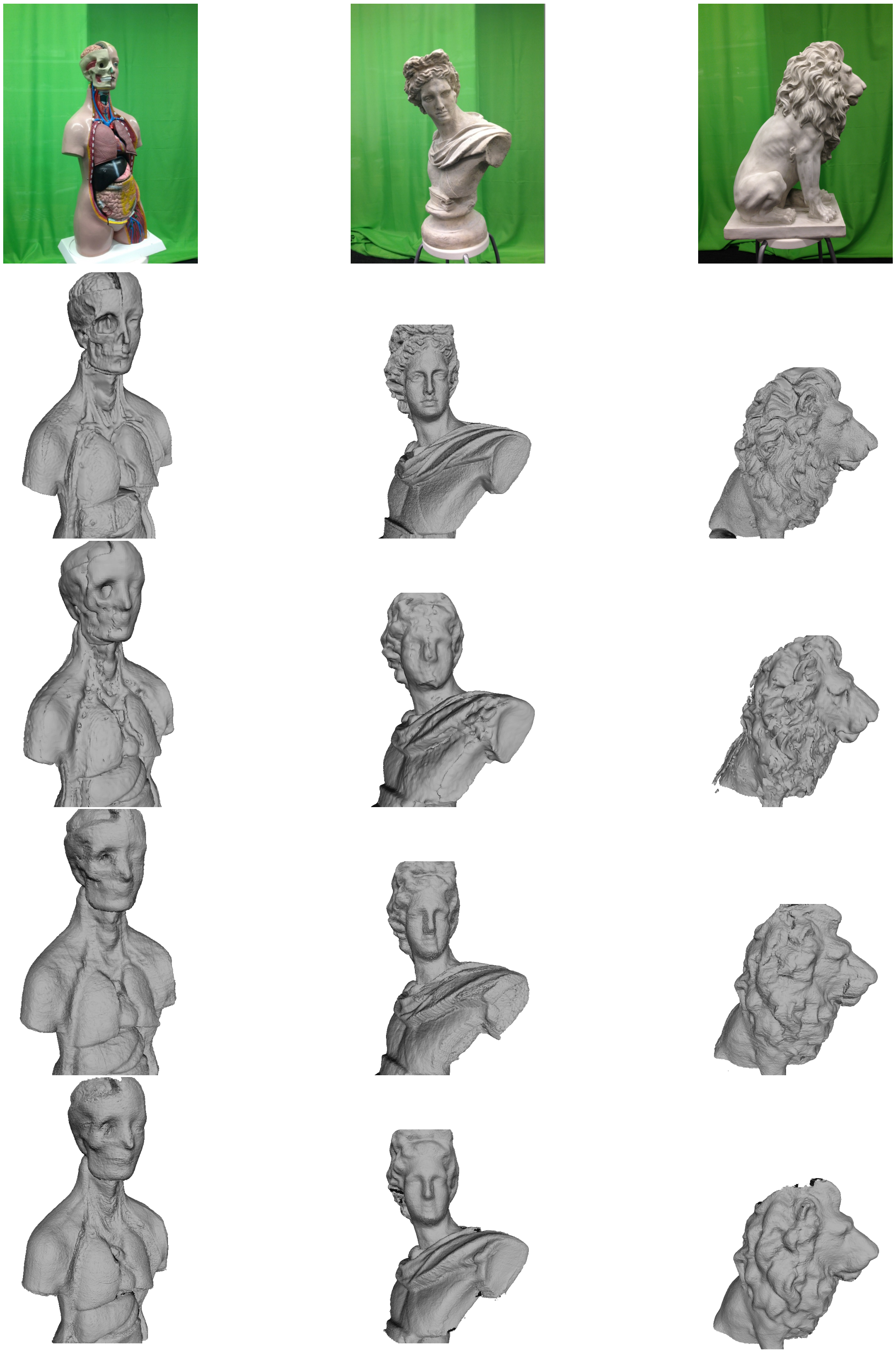
(b) Input disparity map

(e) Downsampled by 5×

(c) Refined disparity map

(f) Downsampled by 9×

Figure 5.4: Depth enhancement comparison of average percentage of bad pixels using Middlebury dataset.



Anatomy *Apollo* *Lion*

Figure 5.5: Reconstructed mesh model comparisons. First Row: real statue model, Second Row: Laser scan model, Third Row: SCENECT, Fourth Row: KinectFusion, and Fifth Row: Proposed method.

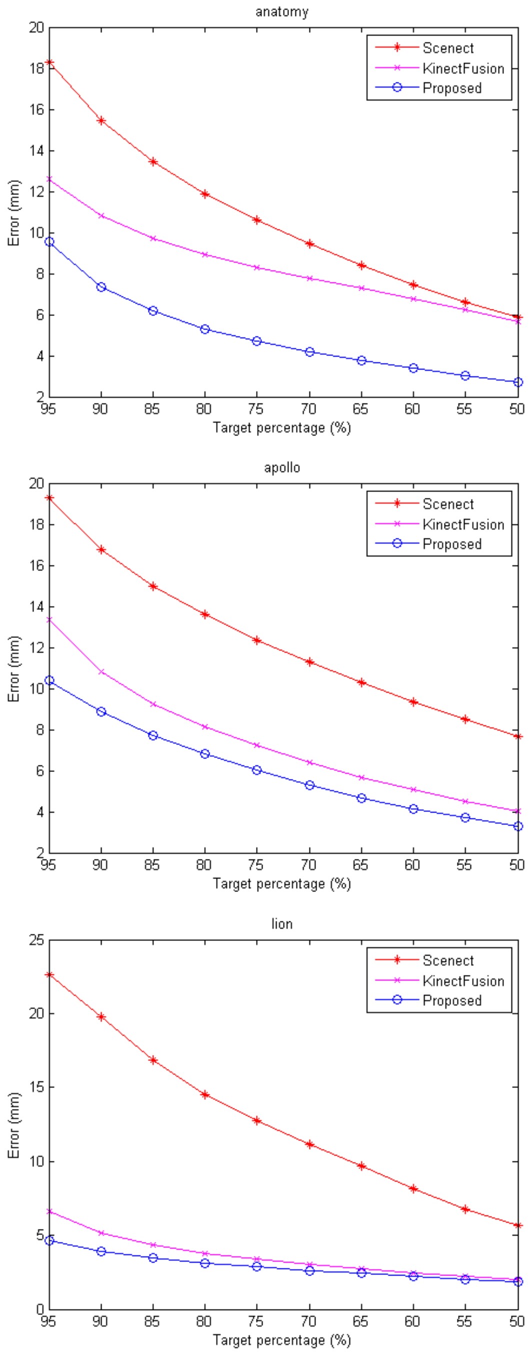


Figure 5.6: Quantitative evaluation results on *anatomy*, *apollo*, and *lion*. The y-axis is the *accuracy* and the x-axis is the target percentage.

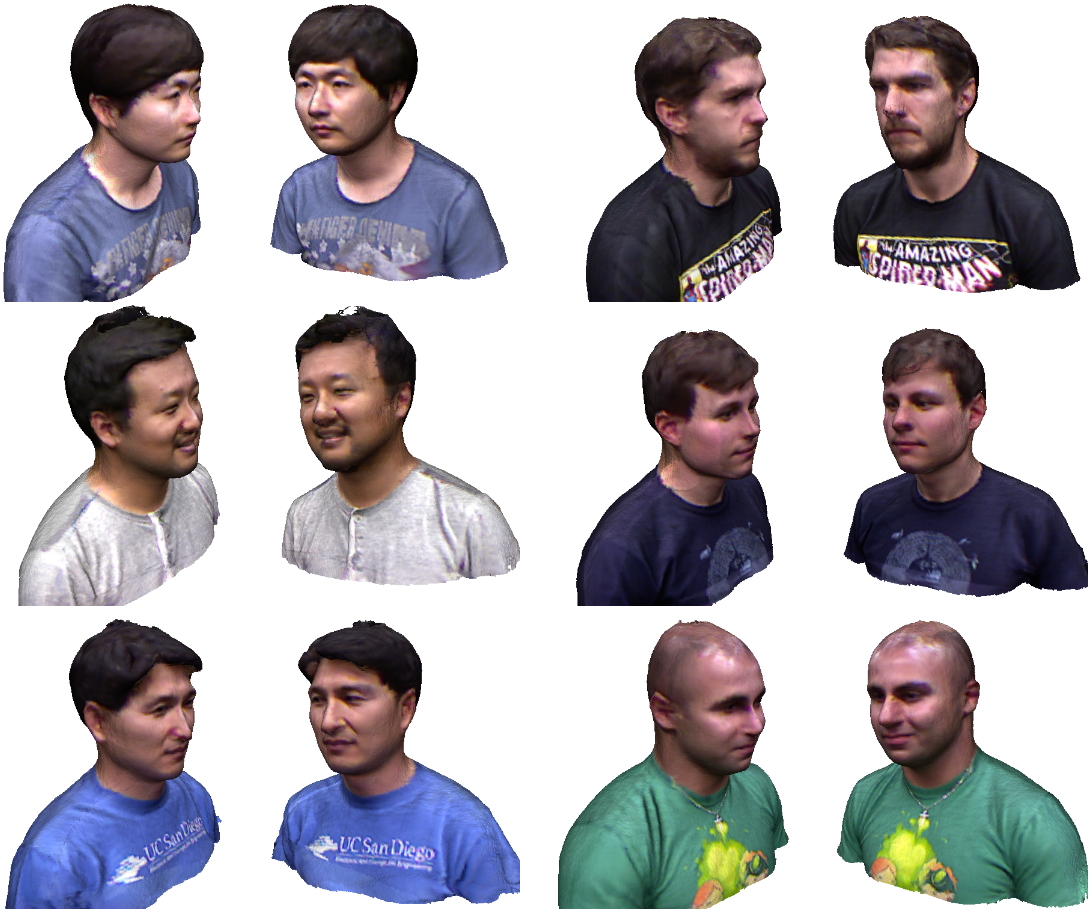


Figure 5.7: Results created with the proposed method, shown as color textured meshes. All results were computed from a handheld camera which captures depth and color images. Note that the mesh models are rendered from two view points.

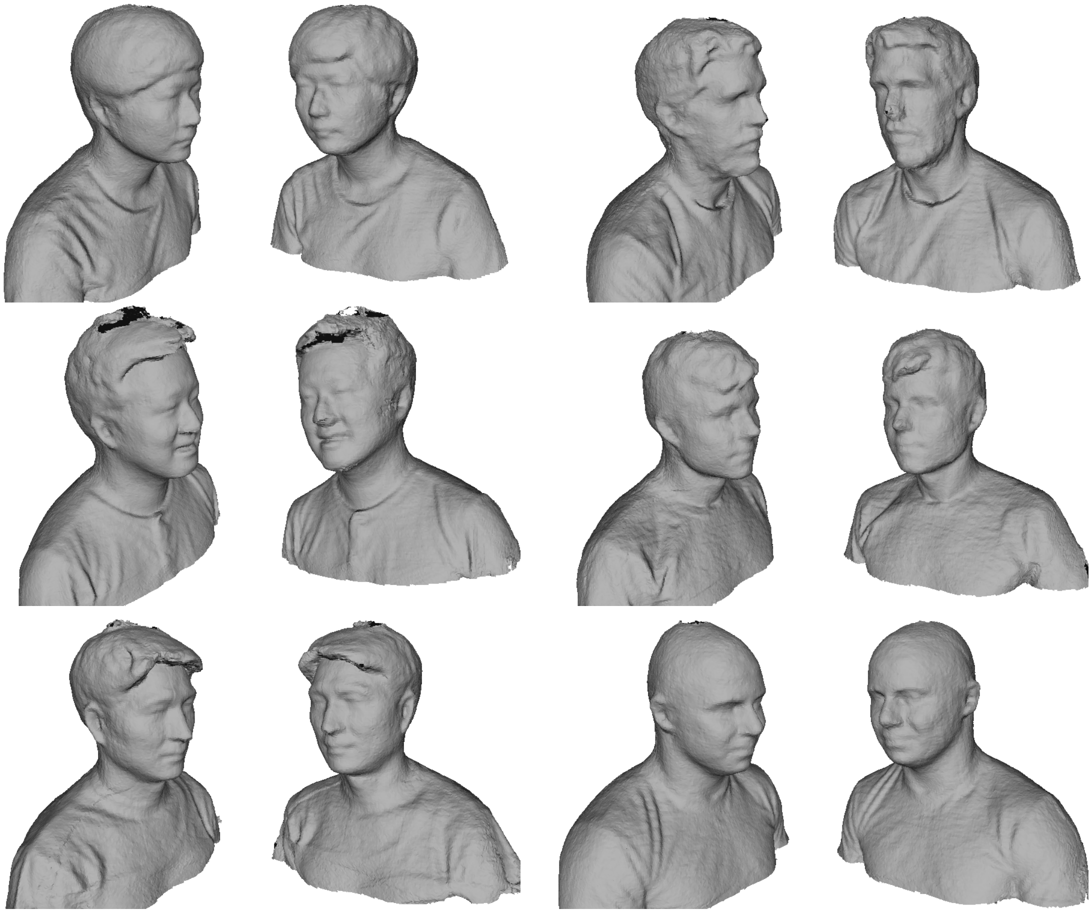


Figure 5.8: Results created with the proposed method, shown as shaded meshes. All results were computed from a handheld camera which captures depth and color images. Note that the mesh models are rendered from two view points.

Table 5.1: Statistics for the reconstruction of *anatomy* dataset at various resolutions.

Voxel Resolution $(voxel/m)$	Running Time (ms)	Mesh Extract. Time (ms)	# Triangles
512	29.5	547	205K
416	25.6	360	135K
256	19.8	127	50K

to the low-quality of depth data and inaccuracy of camera tracking. Figures 5.7 and 5.8 demonstrate the results, shown as color textured and shades meshes. The proposed method achieves accurate color texture reconstruction, since depth data is successfully aligned with color information in the surface refinement process.

We calculate the *accuracy* metric for quantitative comparisons. The *accuracy* represents the distance (error) in which a given percentage of the reconstruction is within the distance from the ground truth model [50]. To generate a ground truth model, we use FARO Focus3D scanner[4] which generates accurate 3D model with upto $\pm 2mm$ distance accuracy. Figure 5.6 shows the statistical accuracy of different target percentages. This figure proves that our system produces more accurate 3D models than the other methods. Input RGB-D video and resulting mesh models can be found at: http://videoprocessing.ucsd.edu/ ultralkl/projects/RealisticReconstruction/

5.6.3 Processing Time

The computation time mostly depends on the number of 3D points and the global volume resolution. Since the 3D point cloud is converted from a depth map, and depth map size is fixed, data acquisition and tracking time are almost constant for each frame. To speed up the camera tracking process, we do not use our initial estimation in SRMF by assuming that we move the camera slowly and steadily. In our system, the reconstruction quality is dependent on the resolution of the volume. The larger the volume, the more details about an object that can be stored, resulting in a more accurate reconstruction. However, the larger global volume size causes the global volume integration and visualization stages to take longer. The results in this chapter were generated

[4] http://www.faro.com/en-us/products/3d-surveying/faro-focus3d/overview

in a volume size $512 \times 512 \times 512$. We tested with different voxel resolutions on the *anatomy* dataset. Table 5.1 summarizes the results of the reconstruction of our dataset at various resolutions. A higher voxel resolution increases the sampling rate, and this improves the reconstruction quality. A reconstruction with the lowest voxel resolution (256 $(voxel/m)$) can be generated in $19.8ms$, and it can produce a mesh model with $50K$ triangles. On the other hand, at the highest voxel resolution $(512$ $(voxel/m))$, the reconstruction is done in $29.5ms$, and produces a mesh model with $205K$ triangles.

5.7 Conclusion

In this chapter, we have proposed a novel approach for realistic surface geometry reconstruction using RGB-D images. In our reconstruction system, high quality color data is acquired by attaching a HD RGB camera on top of the Kinect. This allows us to refine surface geometry using the high frequency features of color images. Using weighted ICP for camera tracking, we achieved better accuracy in the global alignment of scans. The experimental results demonstrate that the proposed refinement method significantly enhances visual appearances and outperforms other state-of-the-art methods.

5.8 Acknowledgments

This chapter is in part a reprint of the following publication: Kyoung-Rok Lee, Truong Q. Nguyen, ”Realistic Surface Geometry Reconstruction Using A Handheld RGB-D Camera”, *Machine Vision and Applications*, (submitted), 2014.

Chapter 6

A Method of 3D Reconstruction of Dynamic Objects

In this chapter, we propose a real-time 3D reconstruction method which generates accurate surface model of dynamic scenes using multiple Kinect cameras. We extend KinectFusion method [1] to reconstruct dynamic objects at each frame. One of the drawbacks of volumetric representation in KinectFusion is that it fails to represent object's surface accurately when the number of views is small. In this work, we present a new depth synthesis technique and a reliability-based weight function to overcome the drawback. The contribution of this chapter is summarized as follows.

- Depth Synthesis: We propose a depth synthesis technique which generates synthesized depth maps at virtual view points by projecting 3D point cloud into 2D image plane. The synthesized depth maps densify the point cloud to cover some surfaces that may be inaccessible to the cameras. Also, a de-noising filter is applied to synthesized depth maps to remove noise caused by properties of the object surface, lighting conditions, or systematic errors.

- Reliability-based Weight Function: We propose a new weighted volumetric representation method to avoid inaccurate surface information in the area where variance of the surface normals is high. We calculate reliability-based weight for each point based on a criteria that depth values with high variance in their neighborhood or depth values along edges are not appropriate for volumetric representation. This weight is used in integration process to attenuate the artifacts of unreliable depth measurements.

- Quantitative Evaluations: We evaluate the proposed method using real-life datasets which include different statues and real people. We compare our results with existing methods by calculating distance error against laser scanned models. We also show accuracy of our proposed method by comparing our reconstruction results with biometric measurements on the corresponding real people.

The rest of this chapter is organized as follows. In Section 6.1, previous methods in dynamic scene reconstruction are described. Then, we give a description of the system setup and overview of the proposed reconstruction system in Section 6.2. We discuss details of the proposed reconstruction method including offline calibration process and online reconstruction process in Section 6.3. In Section 6.4, we provide evaluation on various datasets and analyze the performance of the proposed system. Summary is presented in Section 6.5.

6.1 Related Work

In this section, we discuss recent 3D reconstruction methods that are designed for capturing dynamic scenes or objects using multiple cameras. In recent years, there has been an increasing interest in real-time dynamic scene reconstruction systems for video conferencing, augmented reality, and markerless motion capture. Hasenfratz et al. [51] proposed a real-time full body interaction system using multiple RGB cameras. In their system, full body models are reconstructed using visual hull carving technique and integrated in a virtual environment. Vasudevan et al. [52] proposed a 3D teleimmersive system using multi-view stereo technique. After extrinsic calibration and rectification, a triangular mesh is firstly constructed and disparities of the vertices of the mesh are estimated using a version of normalized cross correlation to reconstruct 3D data.

Several systems achieve 3D reconstruction using range sensors. Kainz et al. [53] presented OmniKinect system which extends KinectFusion to accommodate multiple Kinects simultaneously. In their work, an additional histogram volume is introduced to filter outlier measurements. Also, silhouette carving technique using background subtraction based on color and depth values is exploited for precise edges. Alexiadis et al. [54] presented a multiple Kinects capturing system to generate 3D mesh model of moving foreground objects. In their method, separate meshes is firstly generated by aligned point cloud from each camera. Then, redundant triangles in mesh pairs are removed and meshes are stitched together while removing artifacts in the overlap of the meshes. As a further improvement upon this method, a real-time reconstruction method with volumetric representation is proposed to improve the visual quality of the textures in 3D models [55]. In their volume calculation, they assigned a weight to each voxel based on confidence map to suppress errors that exist on object's boundaries and the surfaces that are perpendicular to the viewing angle. Maimone et al. [56] proposed telepresence system using Kinects, which is designed for personal autostereoscopic telepresence using eye tracking. In their system, depth and color data are merged using photometric constraint to obtain realistic synthesized views based on the location of user's eye.

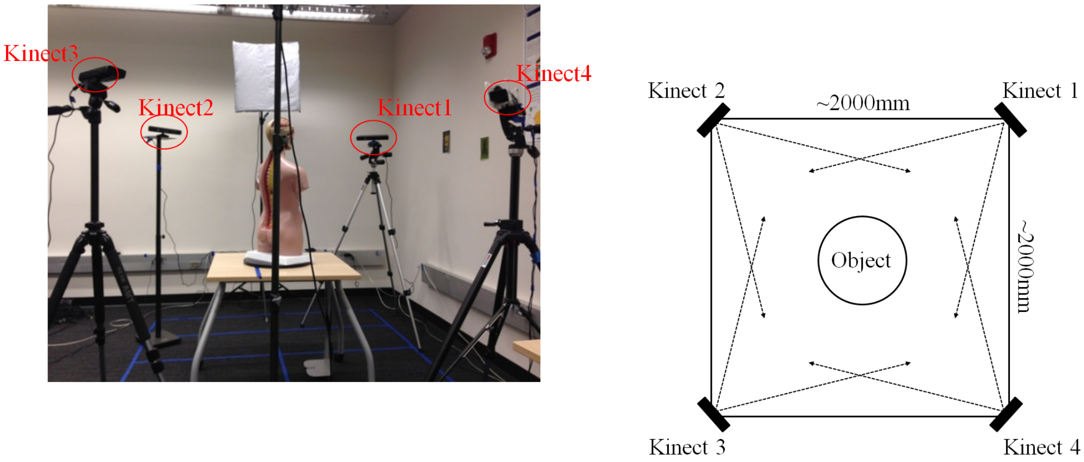
The performance and accuracy of the reconstruction model for these methods are not presented, as no comparison is made with high-resolution scan. In this chapter, we present an accurate dynamic scene reconstruction system using multiple Kinect cameras. We provide accuracy metrics on our benchmark datasets acquired via a laser scanning process to prove accuracy and robustness of the proposed method.

6.2 Overview of the system

The goal of our 3D reconstruction system is to generate geometrically accurate full 3D surface model of real-life dynamic scenes or objects in real-time. In this section, we introduce hardware layout of the proposed system and overview of the proposed reconstruction method.

6.2.1 Kinect Device and System Setup

Our full 3D reconstruction system consists of four Kinect cameras as shown in Figure 6.1(a). We place the cameras on tripods which are located at the corners of a $2m$ x $2m$ square. The top view of the layout is shown in Figure 6.1(b). All cameras



(a) (b)

Figure 6.1: Camera setup.

are connected to a single computer with Intel i7 processor and 12GB RAM with CUDA enabled NVidia GTX Titan. Our system is designed to capture objects which are as large as human upper body. The capturing region can be changed by repositioning the cameras on a larger square. The raw depth and color data are obtained using OpenNI library. In [57, 58, 53], vibrating motors are utilized to reduce the effects of interference between cameras. Unlike these systems, we do not use vibrating motors since the overlapping area is small and interfere between Kinect cameras is trivial. Other literature also shows that interference issues are not as severe as expected [54].

6.2.2 Dynamic Scene Reconstruction System Overview

Figure 6.2 shows an overview of our reconstruction method. Our approach includes offline external camera calibration process and online reconstruction process. The online process consists of data acquisition, depth synthesis, weighted data integration, and visualization. First, depth and color data is captured from multiple Kinect cameras and converted into 3D point clouds and transformed using extrinsic camera parameters from calibration process. Next, we generate synthesized depth maps between depth maps from each camera by setting virtual cameras located between actual cameras and projecting the point cloud into the image plane at a virtual camera position. The

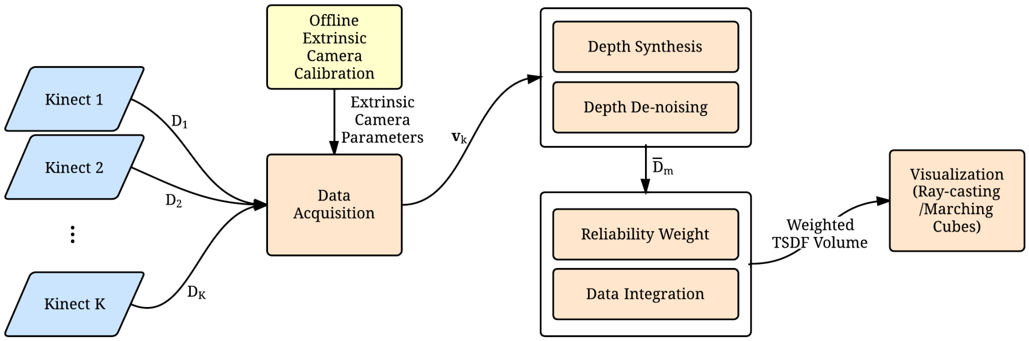


Figure 6.2: Flowchart of the proposed dynamic reconstruction system.

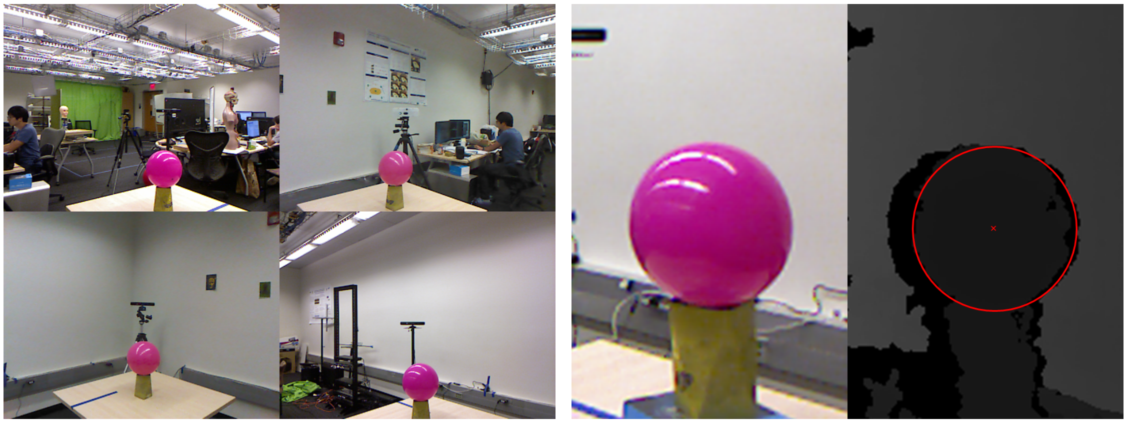
synthesized depth maps densify point cloud to cover surfaces where points are sparsely populated. We also apply a de-noising filter to eliminate outliers and noises in the depth maps. Then, we determine reliability of each depth value based on a criteria that depth values with high variance in their neighborhood or depth values along edges are not appropriate for volumetric representation. The depth information is integrated into a single global volume based on the reliability-based weight. Finally, we visualize the volume as a rendered image via ray-casting or mesh models using marching cubes method.

6.3 Real-Time Dynamic Scene Reconstruction

The proposed approach consists of offline calibration and online surface reconstruction processes. In this section, we describe the details of these processes and describe our contributions.

6.3.1 Camera Calibration

The Kinect has a factory calibration stored onboard, based on a high-level polynomial warping function for registering the depth images (taken by the IR camera) to the RGB images. The OpenNI driver uses this calibration for undistorting the images, which leads to a 1:1 correspondence between pixels in the depth map and the color



(a) (b)

Figure 6.3: Calibration process to estimate extrinsic parameters between Kinects. (a) Captured color data from multiple Kinect cameras (b) Estimated sphere and its center point.

image.

For extrinsic calibration between Kinect cameras, we use a moving sphere as our calibration target. We follow [59], but instead of using RGB data to extract the sphere center in 2D image space, we seek the center from depth data by estimating sphere equation using a sphere fitting algorithm [60]. Specifically, the extrinsic calibration process can be summarized as follows

1. We place an unknown-sized sphere in the capturing space. User interaction is required to select the sphere in image plane. Given known intrinsic parameters of the cameras, we achieve 3D point cloud of the sphere using its depth values and the intrinsic parameters.

2. Given 3D point cloud, we estimate sphere parameters using a sphere fitting algorithm [60]:

$$x^2 + y^2 + z^2 + ax + by + cz + d = 0$$

The center point and radius of a sphere can be extracted by estimating the coefficients of the equation a, b, c, and d. We use the center point as a corresponding point across all Kinect cameras.

3. We move the sphere and repeat steps 1 through 3 until we achieve enough number of corresponding points.

4. Using the corresponding points (a set of center points), we find the relative transformations between prime Kinect camera and other Kinect cameras using quaternion-based orientation estimation method [34].

Figure 6.3(a) shows RGB images captured from four Kinect cameras. The extracted center point and contour of the estimated sphere are shown in Figure 6.3(b). After extrinsic calibration, we acquire a six-degrees-of-freedom (6DOF) camera transformation matrix for each camera, which transforms point cloud in k-th camera space into the first (prime) camera space [34]. We denote the matrix as $\mathbf{T}_{k\to 1}$, consists of a rotation matrix and a translation vector, i.e. $\mathbf{T}_{k\to 1} = [\,\mathbf{R}_{k\to 1} \mid \mathbf{t}_{k\to 1}\,]$.

6.3.2 Data Acquisition

We denote D_k, $k = 1, ..., K$ as the captured depth data from the k-th camera C_k, where K is the total number of cameras in the system. The value of depth map D_k at pixel $\mathbf{p} : [\,u \;\; v\,]^\top$ is represented as $D_k(\mathbf{p})$.

We convert each pixel $\mathbf{p}$ in the depth data into a 3D point, by denoting $\mathbf{v}_k^{cam}(\mathbf{p})$ as the corresponding 3D point in camera space for a pixel $\mathbf{p}$ in the depth data. Given the intrinsic camera calibration matrix $\mathbf{K}$ which is defined in OpenNI, every 3D point $\mathbf{v}_k^{cam}(\mathbf{p})$ in camera space is computed by the back projection function $\mathrm{BProj} : [u \;\; v]^\top \in \mathbb{R}^2 \to [\,x \;\; y \;\; z\,]^\top \in \mathbb{R}^3$:

$$\mathbf{v}^{cam}(\mathbf{p}) = \mathrm{BProj}(\mathbf{p}) = D(\mathbf{p})\mathbf{K}^{-1}\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{6.1}$$

We also denote the global 6DOF camera transformation matrix of camera k as $\mathbf{T}_k = [\,\mathbf{R}_k \mid \mathbf{t}_k\,]$, which transforms the acquired point cloud in camera space into global space. Without loss of generality, we assume that the prime camera coordinate system coincides with the global coordinate system, i.e., $\mathbf{R}_1 = \mathbf{I}$ and $\mathbf{t}_1 = \mathbf{0}$ which leads to a simpler notation of the transformation matrix of each camera $\mathbf{T}_k = \mathbf{T}_{k\to 1}$. Finally, the 3D point $\mathbf{v}_k$ in the global coordinate system can be computed by the transformation

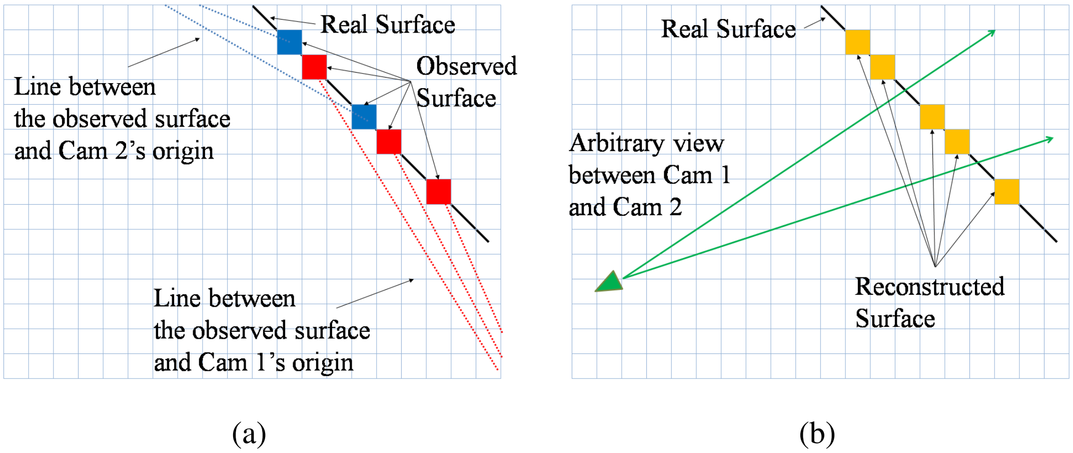

Real Surface
Observed Surface
Line between the observed surface and Cam 2's origin
Line between the observed surface and Cam 1's origin
(a)
Real Surface
Arbitrary view between Cam 1 and Cam 2
Reconstructed Surface
(b)

Figure 6.4: Visualization of TSDF when surface points are sparsely populated. (a) Points in the area where their surface normal is almost orthogonal to camera's origin are sparsely populated. (b) TSDF in some voxels on the real surface cannot be computed and leads to holes.

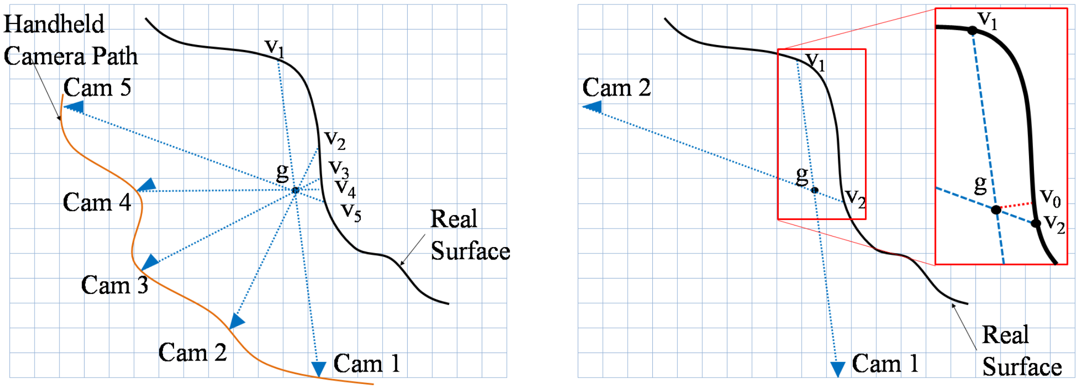
matrix $\mathbf{T}_k$ as:

$$\mathbf{v}_k(\mathbf{p}) = \mathbf{T}_k \mathbf{v}_k^{cam}(\mathbf{p}) \tag{6.2}$$

6.3.3 TSDF Volumetric Representation

The volumetric representation method using cumulative truncated signed distance functions (TSDF) is proposed by Curless and Levoy [36]. Because of certain desirable features such as robustness in the presence of outliers by incremental updating and the ability to fill missing data, it is widely used in 3D reconstruction methods [1, 61, 53, 55]. In volumetric representation, each volume element, voxel, stores a signed distance from the voxel to the nearest object's surface; it has a negative value when inside the surface, a positive value when outside the surface, and 0 when on the surface. In [1], TSDF is exploited to fuse live range images incrementally in real-time. Given the estimated camera pose, the geometric information of each frame is transformed into the global space and projected into global volume space.

However, the TDSF technique is not always the most suitable method for high-quality reconstruction when the number of views is small. There are two major drawbacks.



(a) (b)

Figure 6.5: Visualization when TSDF fails to represent the correct distance to surface. (a) In [1], TSDF of $\mathbf{g}$ is determined by Cam 1 to 5. (b) TSDF of $\mathbf{g}$ is determined by average of distances between $\mathbf{g}$ and $\mathbf{v}_1$ to $\mathbf{v}_2$, which is not similar to the actual distance to $\mathbf{v}_0$.

- To reconstruct dynamic objects at each frame, we only use a few data points (K= 4), with small overlap regions. Therefore, we may have very sparsely populated points in the area where its surface normal is almost perpendicular to camera's origin (See Figure 6.4(a)). Since TSDF is only caculated for voxels which are located on the line between the observed surface point and camera's origin, the sparsity of the point cloud leads to holes in surface reconstruction. Figure 6.4(b) shows the case where TSDF cannot be computed in some voxels on the real surface.

- In [1], plenty of frames are fused into global volume to represent object's surface. Figure 6.5(a) demonstrates the visualization of TSDF determined by depth data from camera 1 to 5. TSDF of voxel $\mathbf{g}$ is computed by averaging distances between the voxel and observed surface points ($\mathbf{v}_1$, ..., $\mathbf{v}_5$). However, the volumetric presentation with TSDF becomes significantly inaccurate when the number of measurements is small and some of the TSDF values fail to represent the correct distance to object's surface. Figure 6.5(b) demonstrates the problem. The TSDF of voxel $\mathbf{g}$ is determined by average of distances between the voxel and $\mathbf{v}_1$ to $\mathbf{v}_2$, which is not similar to the actual distance to surface $\mathbf{v}_0$.

In our approach, we introduce a new depth synthesis method and reliability-based weight in integration process to overcome the drawbacks for high-quality 3D reconstruction.

6.3.4 Depth Synthesis and Depth De-noising

Depth Synthesis

To generate synthesized depth maps, we first create virtual cameras that are located between real cameras. We assume a circular camera path based on locations of real cameras. The center point $\mathbf{c}$ and radius r of the circle are calculated as:

$$\mathbf{c} = [\, x^{center} \;\; y^{center} \;\; z^{center} \,]^\top = \sum_{k=1}^{K} \mathbf{t}_k / K$$
$$r = \sum_{k=1}^{K} \|\mathbf{c} - \mathbf{t}_k\| / K \tag{6.3}$$

Now, we find the position and orientation for each virtual camera. We denote $\mathbf{T}_s^{k,k+1} = [\mathbf{R}_s^{k,k+1} | \mathbf{t}_s^{k,k+1}]$, $s = 1, ..., S$ as the transformation matrix of s-th virtual camera between two real cameras C_k and C_{k+1}, where S is the number of virtual cameras between two adjacent real cameras. The position of the s-th virtual camera is determined as:

$$\mathbf{t}_s^{k,k+1} = [\, r\sin(\theta) + x^{center} \;\; 0 \;\; z^{center} - r\cos(\theta) \,]^\top \tag{6.4}$$

where θ is the clockwise angle between the virtual camera and the first camera. i.e., $\theta = -((k-1)\pi/2 + (s/S+1)\pi/2)$. To find the orientation of virtual camera, we convert the orientation of each real camera into its quaternions, which are considered as a better representation for interpolation than Eulerian angles. We denote $\mathbf{q}_k$ and $\mathbf{q}_{k+1}$ as quaternions converted from rotation matrices $\mathbf{R}_k$ and $\mathbf{R}_{k+1}$, respectively. The orientation of virtual camera is determined by interpolating two quaternions using Spherical linear interpolation (Slerp) [62]:

$$\mathbf{R}_s^{k,k+1} = \mathrm{Quat2Rotation}(\mathrm{Slerp}(\mathbf{q}_k, \mathbf{q}_{k+1}, s/(S+1))) \tag{6.5}$$

Figure 6.6 shows orientations and positions of virtual cameras which are projected onto x-z plane when K = 4 and S = 3.

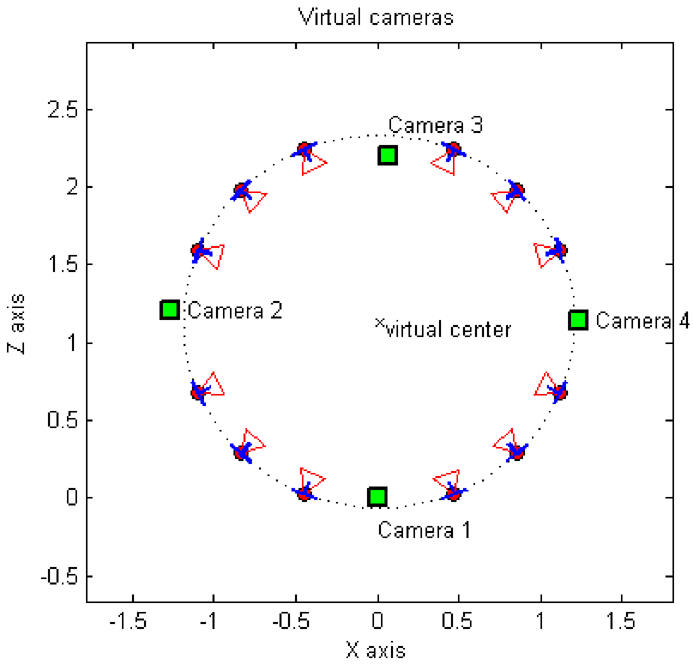


Figure 6.6: Orientations and positions of virtual cameras, which are projected onto the x-z plane (K = 4 and S = 3).

Next step is to generate synthetic depth maps by projecting aligned point clouds into image plane at each virtual camera. We denote $D_s^{k,k+1}$ as the synthesized depth map with the transformation $\mathbf{T}_s^{k,k+1}$. Using the transformation matrix of the virtual camera, we synthesize depth $D_s^{k,k+1}$ by warping both D_k and D_{k+1}. The warping process is done by projecting $\mathbf{v}_k$ and $\mathbf{v}_{k+1}$ into the virtual camera. The projection function $\mathrm{Proj} : [\,x \;\; y \;\; z\,]^\top \in \mathbb{R}^3 \rightarrow [\,u \;\; v\,]^\top \in \mathbb{R}^2$ is defined as:

$$\begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} = \mathrm{Proj}(\mathbf{v}^{cam}) = \frac{1}{z}\mathbf{K}\mathbf{v}^{cam} \tag{6.6}$$

In depth synthesis process, we copy the depth value of projected point $\mathbf{p}'$ to nearby pixels $\mathbf{q}' \in \Omega_{\mathbf{p}'}$ to populate points on the surface where depth data from real cameras cannot cover. $\Omega_{\mathbf{p}'}$ is a small rectangular patch centered at pixel $\mathbf{p}'$ and the size is chosen empirically. We also use *z-buffer* to decide which points are visible and which are hidden; if another 3D point must be projected in the same pixel in the synthesized depth map, their depth values are compared and the closer point to the virtual camera is chosen. Then, the chosen depth is saved to the *z-buffer* by replacing the old one. The process is summarized in Algorithm 2.

Depth De-noising

Errors in range images generated by Kinect sensor include outliers in transition areas and random noise on geometrically flat or smooth surfaces due to properties of the object surface, lighting conditions, or systematic errors. Since our system only uses data from a single shot per camera for dynamic reconstruction, the quality of geometry data is critical to the resulting 3D model. To suppress artifacts from depth noise, we apply a bilateral filter that is composed of spatial and range filters to reduce noise while preserving geometrical discontinuities. The filtered depth value at pixel $\mathbf{p}$ is determined by:

$$\dot{D}(\mathbf{p}) = \frac{\sum_{\substack{\mathbf{q}\notin\Phi,\\ \mathbf{q}\in\Omega_{\mathbf{p}}}} f_r(|D(\mathbf{p}) - D(\mathbf{q})|) f_s(\|\mathbf{p} - \mathbf{q}\|) D(\mathbf{q})}{\sum_{\substack{\mathbf{q}\notin\Phi,\\ \mathbf{q}\in\Omega_{\mathbf{p}}}} f_r(|D(\mathbf{p}) - D(\mathbf{q})|) f_s(\|\mathbf{p} - \mathbf{q}\|)} \tag{6.7}$$

where Φ is the region where Kinect cannot determine depth information. f_r is the range filter and f_s is the spatial filter, which are Gaussian filters with variances σ_r and σ_s, respectively.

We denote a set of filtered real depth $\dot{D}_k$ and synthesized depth $\dot{D}_s^{k,k+1}$ as $\bar{D}_m$, $m = 1, ..., M$ and $M = K * (S + 1)$. Also, we denote a union set of $\mathbf{T}_k$ and $\mathbf{T}_{k+1}^s$ as $\bar{\mathbf{T}}_m$.

6.3.5 Data Integration using Reliability-based Weight

Reliability-based Weight Function

We introduce a new weight function based on reliability of depth value in the integration process. In our approach, the reliability-based weight is determined based on the criteria that depth values with high variance in their neighborhood or depth values along edges are not reliable for the TSDF. The reliability-based weight is determined by the following equation:

$$\alpha_m(\mathbf{p}) = \sum_{\substack{\mathbf{q}\notin\Phi,\\ \mathbf{q}\in\Omega_{\mathbf{p}}}} f_r(|\bar{D}_m(\mathbf{p}) - \bar{D}_m(\mathbf{q})|)/(\text{\# of } \mathbf{q}) \tag{6.8}$$

Algorithm 2 Depth Synthesis

```
input: $D_k, D_{k+1}, \mathbf{v}_k, \mathbf{v}_{k+1}, \mathbf{T}_k, \mathbf{T}_{k+1}, \mathbf{T}_s^{k,k+1}$
output: $D_s^{k,k+1}$
for pixel $\mathbf{p}$ in $\mathbf{D}_k$ do
  if $D_k(\mathbf{p}) \notin \Phi$ then
    $\mathbf{p}' = \mathrm{Proj}((\mathbf{T}_s^{k,k+1})^{-1}\mathbf{T}_k\mathbf{v}_k(\mathbf{p}))$
    for pixel $\mathbf{q}'$ in $\Omega_{\mathbf{p}'}$ do
      if $\mathbf{q}' \in$ [1:width,1:height] & zbuffer$(\mathbf{q}') > D_k(\mathbf{p})$ then
        $D_s^{k,k+1}(\mathbf{q}') = D_k(\mathbf{p})$
        zbuffer$(\mathbf{q}') = D_k(\mathbf{p})$
      end if
    end for
  end if
end for
for pixel $\mathbf{p}$ in $D_{k+1}$ do
  if $D_{k+1}(\mathbf{p}) \notin \Phi$ then
    $\mathbf{p}' \leftarrow \mathrm{Proj}((\mathbf{T}_s^{k,k+1})^{-1}\mathbf{T}_{k+1}\mathbf{v}_k(\mathbf{p}))$
    for pixel $\mathbf{q}'$ in $\Omega_{\mathbf{p}'}$ do
      if $\mathbf{q}' \in$ [1:width,1:height] & zbuffer$(\mathbf{q}') > D_{k+1}(\mathbf{p})$ then
        $D_s^{k,k+1}(\mathbf{q}') = D_{k+1}(\mathbf{p})$
        zbuffer$(\mathbf{q}') = D_{k+1}(\mathbf{p})$
      end if
    end for
  end if
end for
```

$\alpha_m(\mathbf{p})$ in Eq. (6.8) has value close to zero unless at area with geometrically smooth surface. The resulting weight is used in the following data integration step.

Data Integration

Given the camera poses, the surface depth information of the real and virtual views is transformed into the global coordinate space and is projected onto each corresponding voxel in a global volume space with a value determined by the weighted TSDF $\xi(\mathbf{g})$, expressed as:

$$\xi(\mathbf{g}) = \sum_{m}^{M} \alpha_m(\mathbf{p})\psi(\|\bar{\mathbf{t}}_m - \mathbf{v}\| - \bar{D}_m(\mathbf{p}))/\sum_{m}^{M} \alpha_m(\mathbf{p}) \tag{6.9}$$

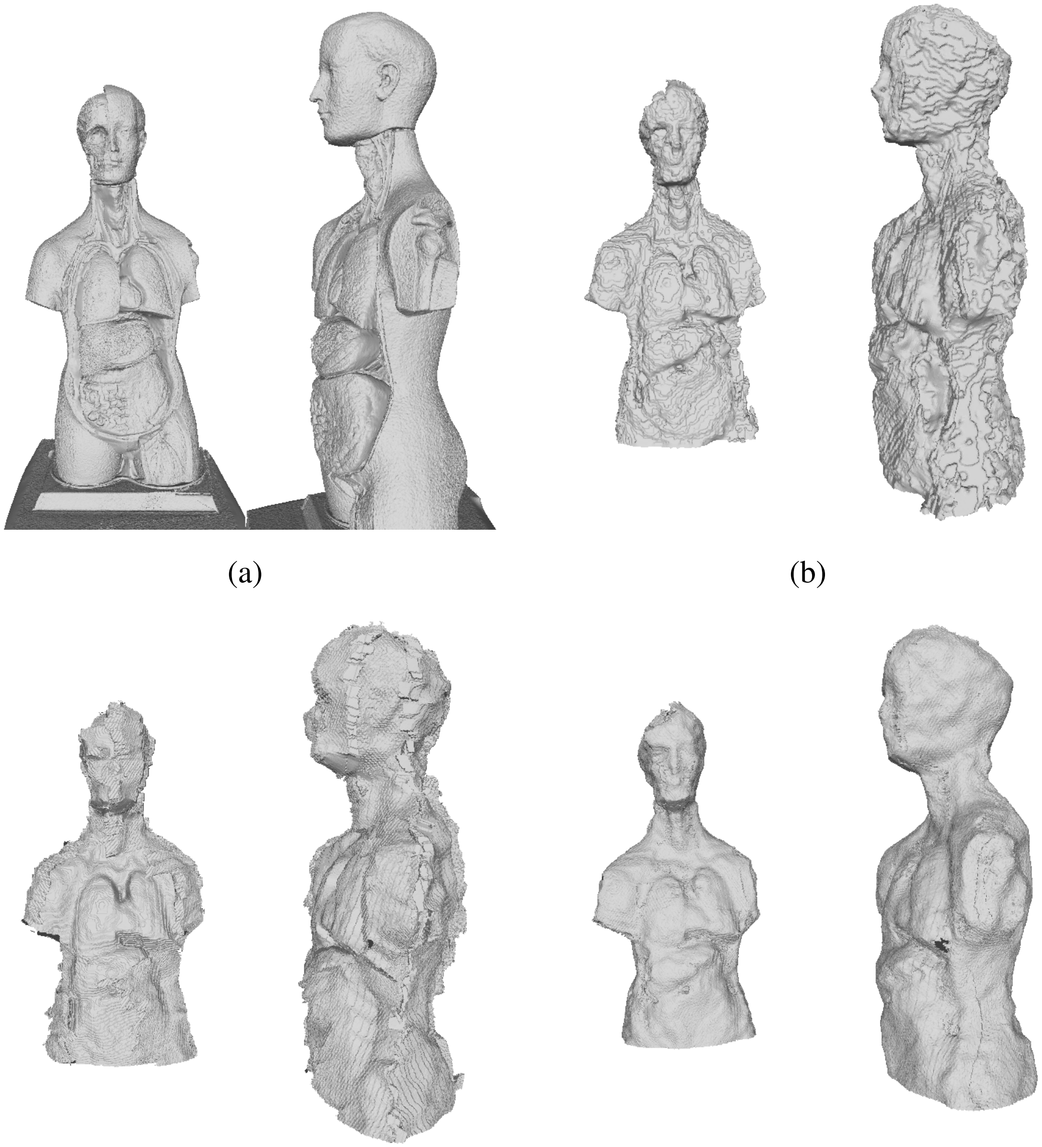
where

$$\begin{aligned} \mathbf{v} &= \zeta(\mathbf{g} + [\,0.5\ \ 0.5\ \ 0.5\,]^\top) \\ \mathbf{p} &= \mathrm{Proj}(\bar{\mathbf{T}}_m^{-1}(\mathbf{v})) \\ \psi(n) &= \begin{cases} \min(1, n/\mu)sgn(n), & \text{if } n > \mu \\ \varnothing, & \text{otherwise} \end{cases} \end{aligned}$$

ζ is the scalar value which represents the voxel size; we assume all sides of voxel has the same length. $\mathbf{v}$ is the global coordinate of voxel $\mathbf{g}$ and $\|\bar{\mathbf{t}}_m - \mathbf{v}\|$ represents the distance from camera center to voxel location in global space. Depth value $\bar{D}_m(\mathbf{p})$ is the distance from camera origin to observed surface. By averaging the values together, we can build accurate 3D representation that is consistent with all views. While the KinectFusion uses constant weight for averaging process, we exploit reliability factor α to avoid possible inaccuracy especially where variance of the surface normals is large.

6.4 Results

In this section, experiments performed to evaluate the proposed system are described. We first compare the accuracy of the proposed method with other state-of-the-art methods. Then, we compare the real human biometric measurements with reconstruction results. Finally, we present computation analysis by presenting execution time and corresponding frame rate.



(a)

(b)

(c)

(d)

Figure 6.7: Reconstructed mesh model comparisons on *anatomy* dataset. (a) Laser scan model, (b) Screened Poisson, (c) KinectFusion, and (d) Proposed method

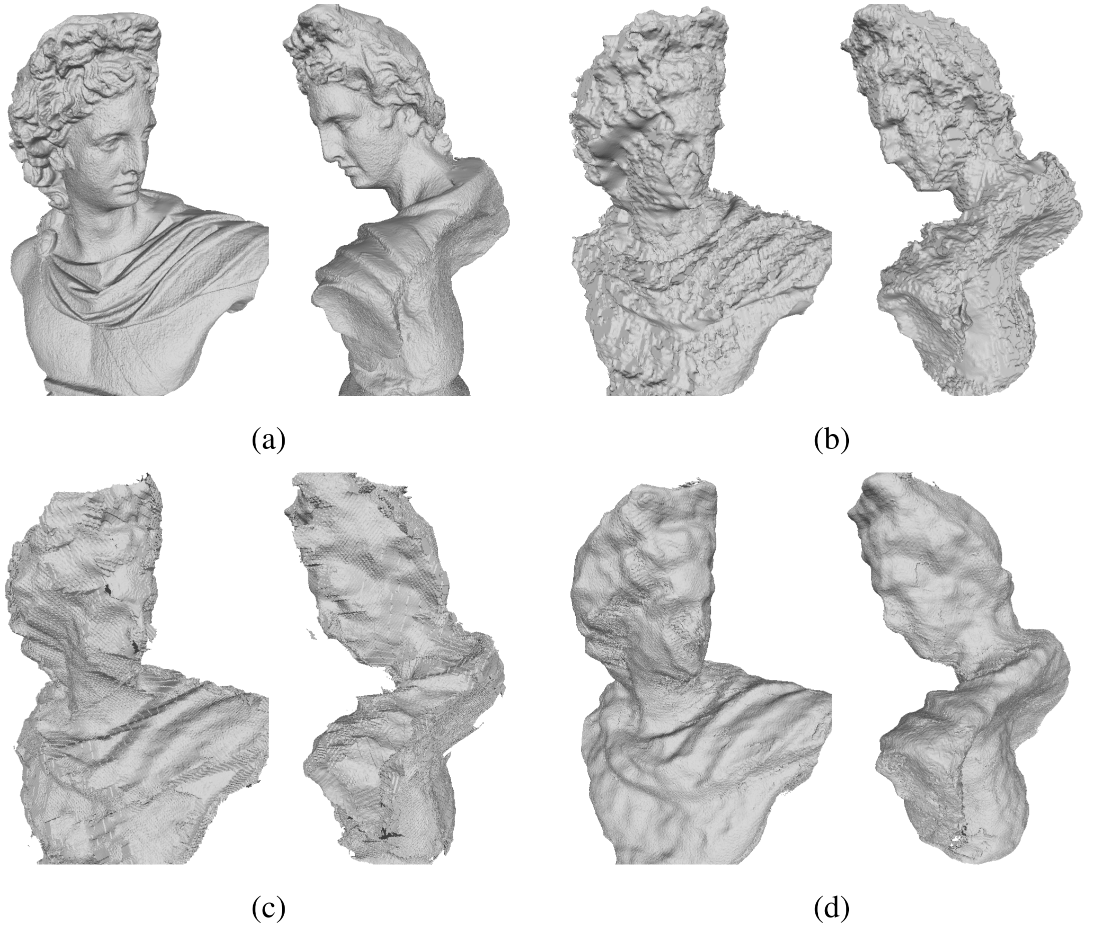
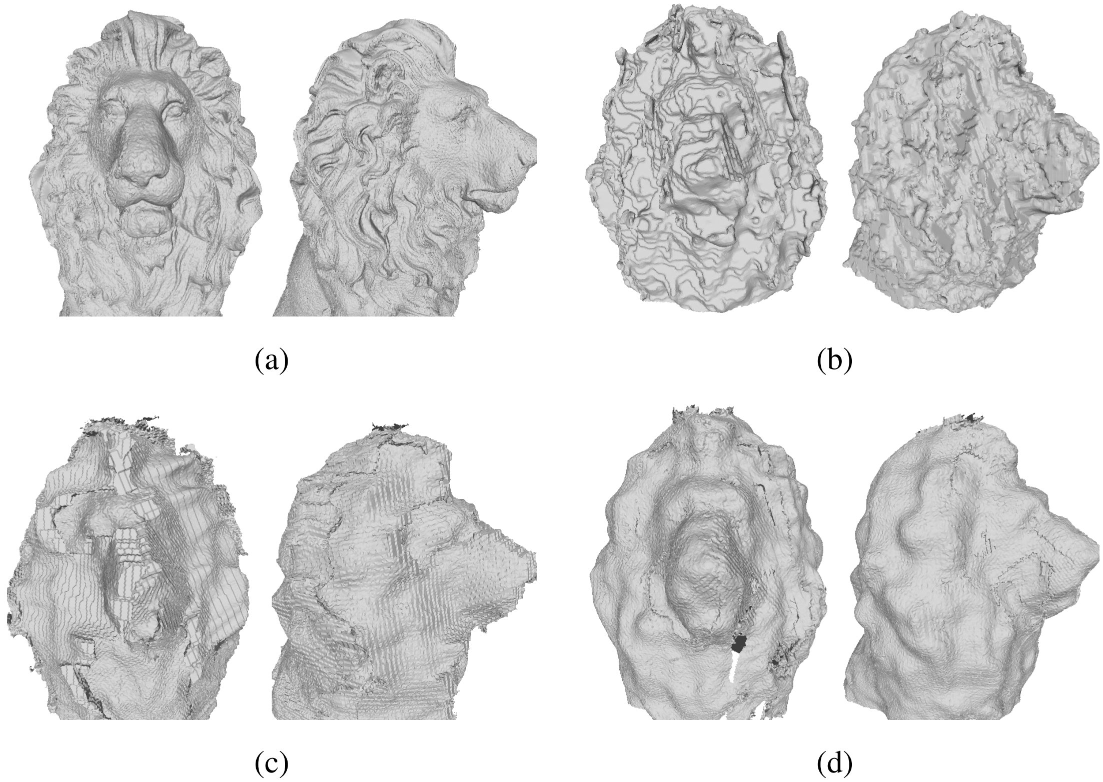


Figure 6.8: Reconstructed mesh model comparisons on *apollo* dataset. (a) Laser scan model, (b) Screened Poisson, (c) KinectFusion, and (d) Proposed method



(a)

(b)

(c)

(d)

Figure 6.9: Reconstructed mesh model comparisons on *lion* dataset. (a) Laser scan model, (b) Screened Poisson, (c) KinectFusion, and (d) Proposed method

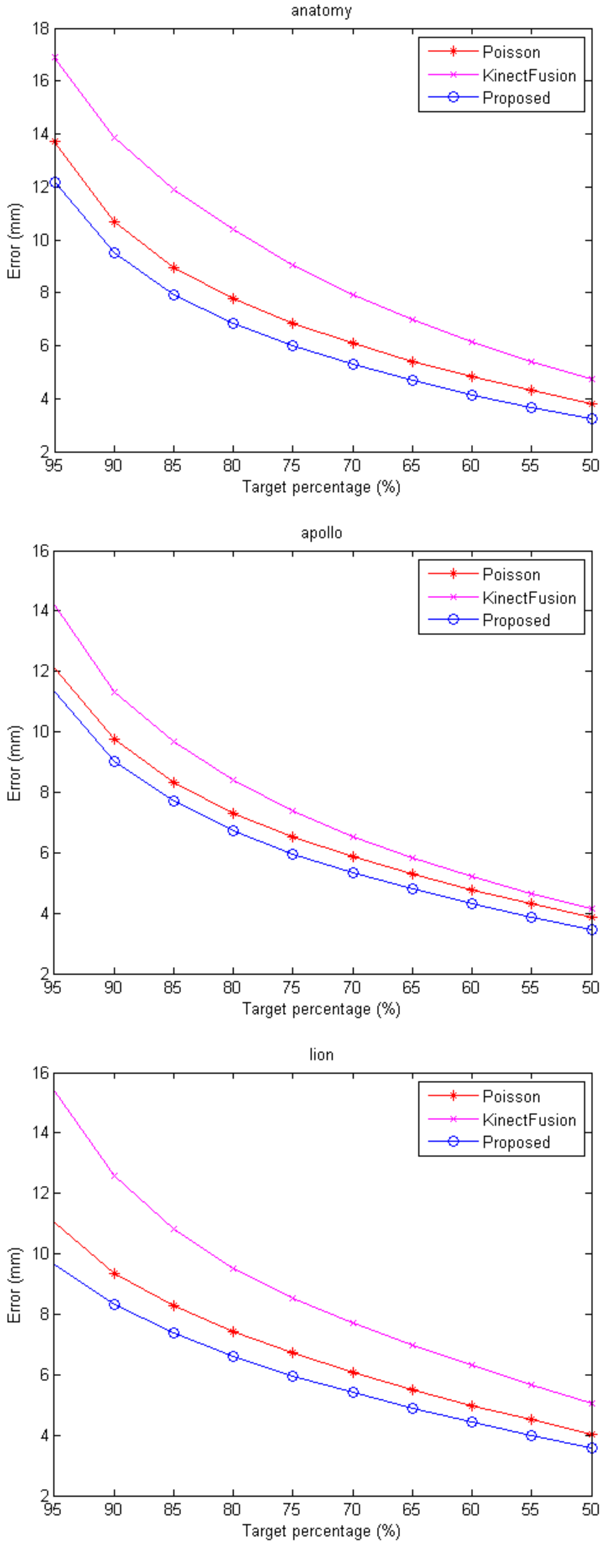


Figure 6.10: *Accuracy* plot. The y-axis is the *accuracy*, and the x-axis is the target percentage. *anatomy* (Top), *apollo* (Middle), and *lion* (Bottom).

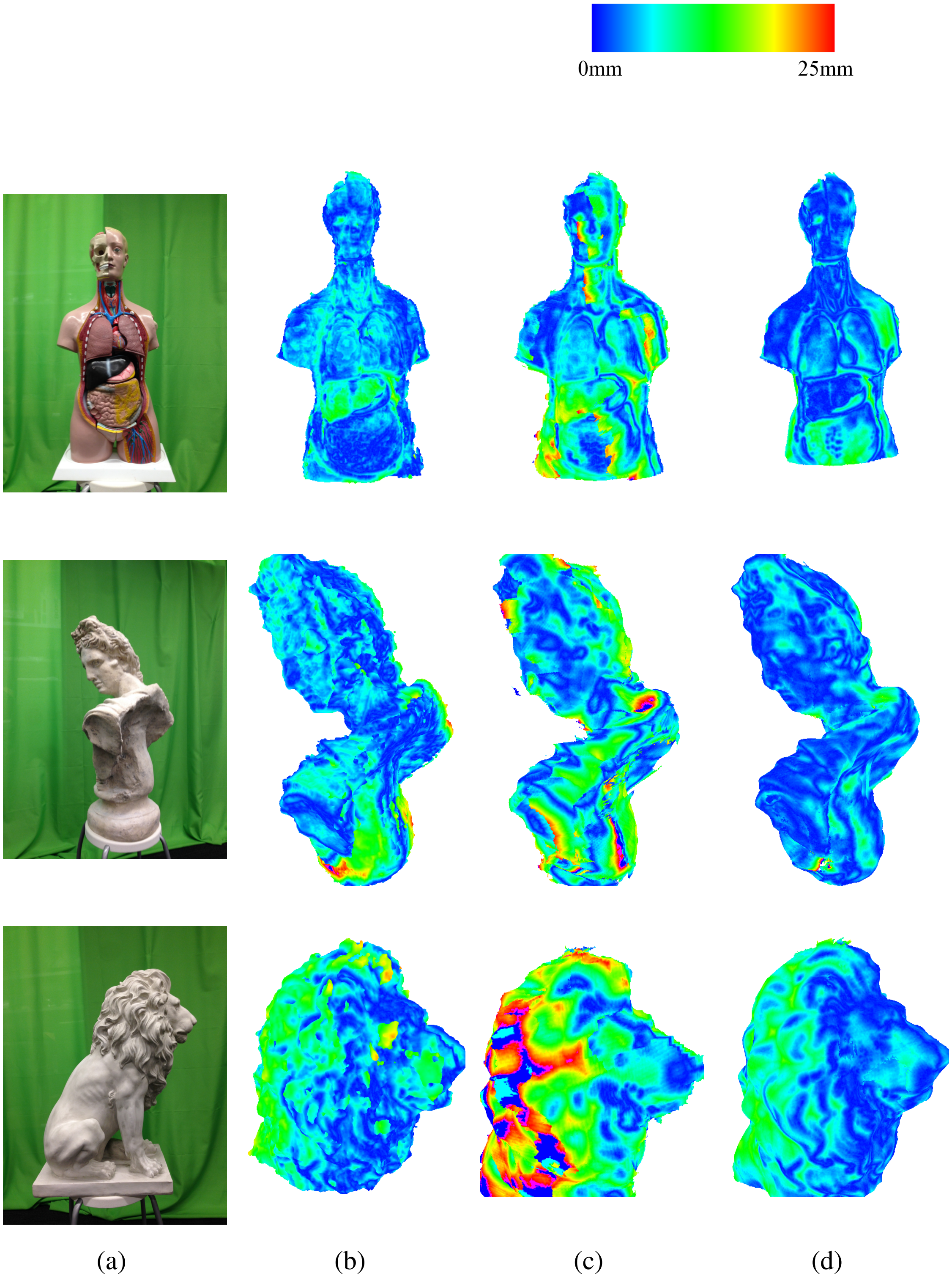


Figure 6.11: Quantitative evaluation results on *anatomy* (Top), *apollo* (Middle), and *lion* (Bottom). (a) Color image, and Color-coded distance errors against laser scan ((b): Screened Poisson, (c): KinectFusion, and (d): Proposed Method)).

6.4.1 Comparative Results

To show our performance, we convert global volume to a mesh model using Marching Cubes method [63] and compare the results with those from laser scanner and state-of-the-art methods KinectFusion [1] and Screened Possion method [49]. The ground truth models are acquired by the FARO Focus3D scanner, which is an accurate laser scanner with up to $\pm 2mm$ distance accuracy. The high quality model is captured with controlled environment for several minutes. Originally, KinectFusion is designed to reconstruct static scene with a handheld camera and requires multiple frames to reconstruct the scene. In our experiments, transformation parameters from extrinsic calibrations are used for frame registration and conventional TSDF is used for integration of data from multiple cameras. Screened Poisson reconstruction method, which is an improved version of Poisson reconstruction method [64], creates watertight surface models from 3D point cloud. The method is superior to [54] in subjective evaluation of resulting mesh quality, but requires a few seconds to create surfaces because of high computational complexity [54].

Figures 6.7, 6.8, and 6.9 show the ground truth and estimated mesh models generated from the proposed method and other methods. As illustrated, results from both KinectFusion and Screened Poisson have significant artifacts in partial overlap of the geometric data caused by inaccurate boundary of the depth map from Kinect camera. In the proposed method, we are able to improve the surface quality by applying de-noising filter at multiple synthesized views. Also, reliability weight successfully attenuates artifacts caused by erroneous depth values with high variance in their neighborhood or depth values along edges.

For quantitative comparisons, we employ *accuracy* evaluation metric, which is the distance (error) such that a given percentage of the reconstruction is within the distance from the ground truth model [50]. Figure 6.10 shows statistic of *accuracy* between reconstructed model and ground truth model for different target percentages. Figure 6.11 shows color-coded distance errors against laser scan. The evaluation results show that our approach outperforms other methods in terms of *accuracy* for the three dataset due to the use of synthesized depth maps and proposed weighted TSDF.

In addition, we evaluate the proposed method in dynamic situations to prove

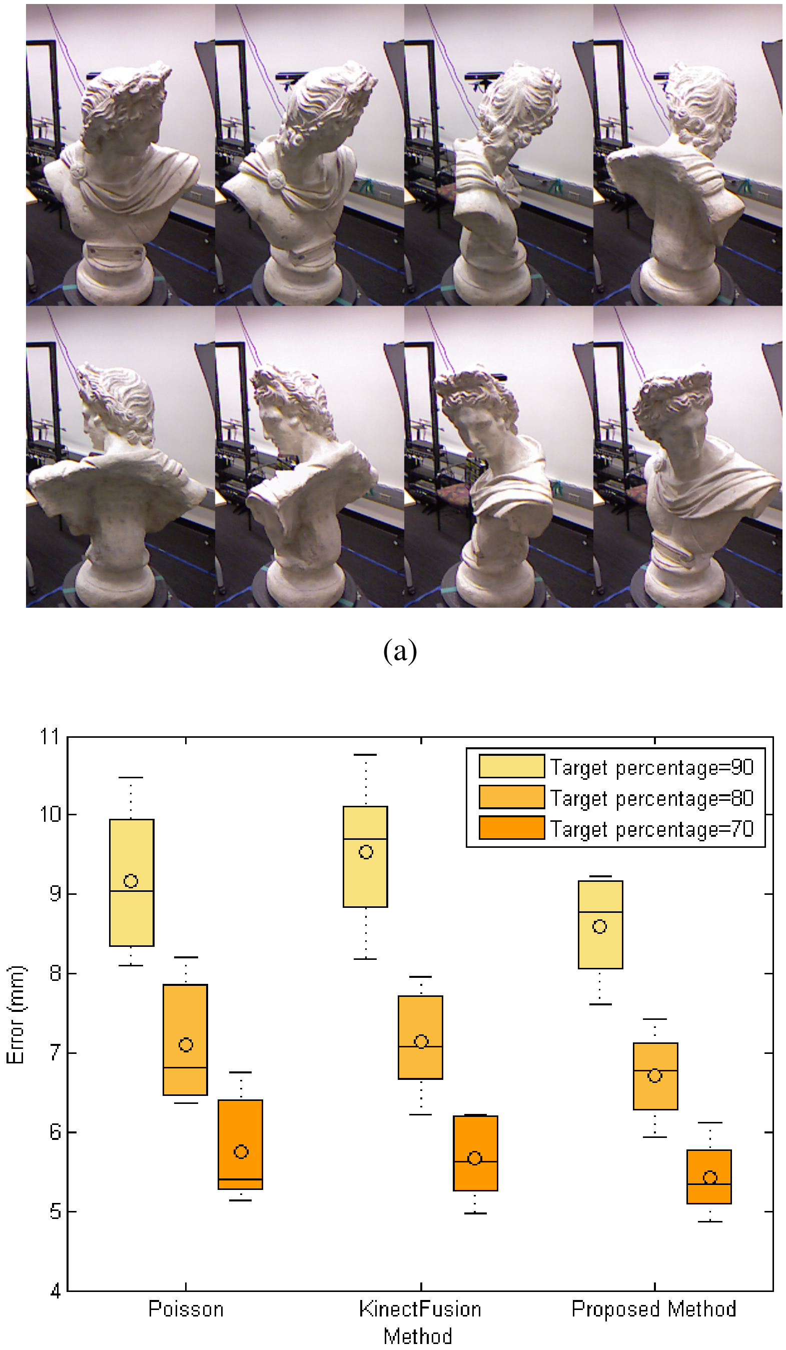
Table 6.1: Biometric measurements on real persons

		Chin to Nose	Shoulder Width	Elbow to Wrist
	Mesh	8.9	42.5	26.8
Byeong	Biometric	9.3	41.6	26.0
	Error (cm)	**0.4**	**0.9**	**0.8**
	Mesh	7.3	36.5	25.1
Haleh	Biometric	7.9	37.0	25.5
	Error (cm)	**0.6**	**0.5**	**0.4**
	Mesh	8.3	42.0	26.7
Kyoung	Biometric	7.9	41.3	25.6
	Error (cm)	**0.4**	**0.7**	**1.1**

robustness. The *apollo* statue model on a turntable is rotated by 45 degrees in the capturing area to capture the same model but with different poses. Eight mesh models with eight different poses are generated to calculate error variances. Figure 6.12 shows the variance plot on generated mesh models. As shown in the figure, the proposed method produces smaller mean error and smaller variance compared to other methods and this verifies that the proposed method is more robust than the others in dynamic situations.

6.4.2 Upper Body Results

Figure 6.13 shows reconstruction results of proposed method on real people. Table 6.1 shows biometric measurements which are obtained by measuring the reconstructed human models and comparing them with the corresponding real people. In this experiments, three biometric measurements are calculated, which are ”Chin to Nose”, ”Shoulder Width”, and ”Elbow to Wrist”. The errors in the table show that our reconstructed models are very accurate (about $1cm$). Unlike [41] which requires people on a turntable should stand still for a few seconds, our system can generate full 3D models using only one frame each from four Kinect cameras.



(a)

(b)

Figure 6.12: Robustness evaluation. (a) Color images of *apollo* rotating on a turntable. (b) Mean-variance plot for *apollo* dataset.

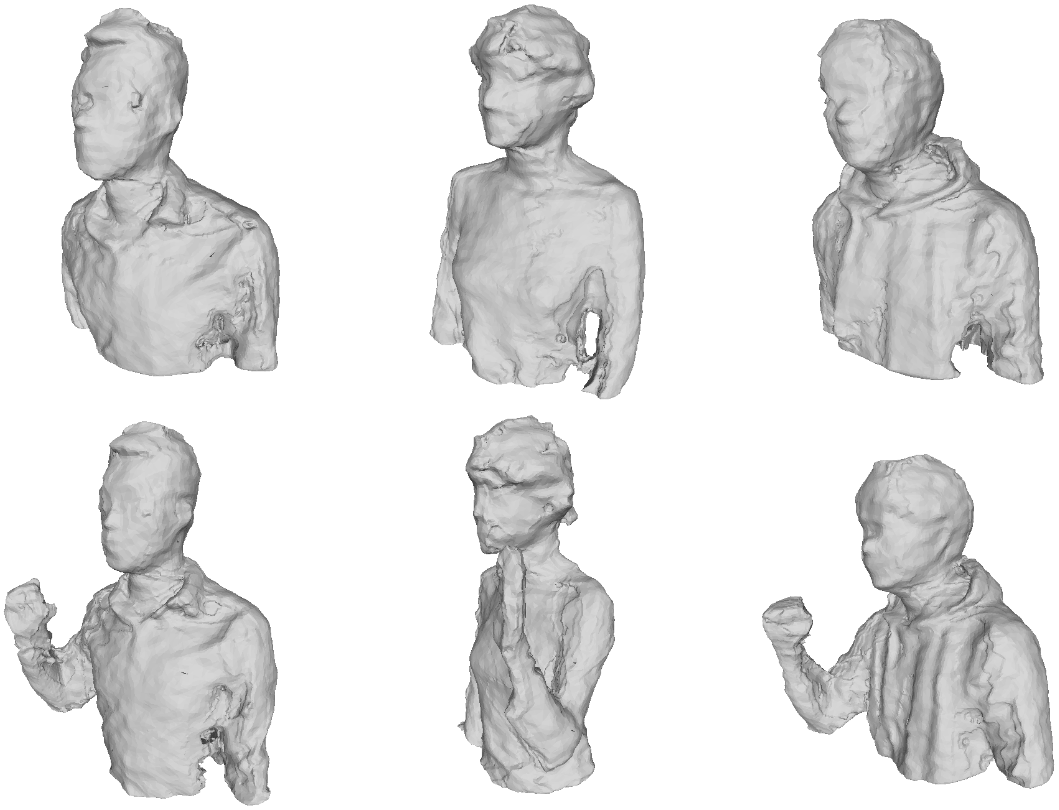


Figure 6.13: Upper Body Results. *Byeong* (Left), *Haleh* (Middle), and *Kyoung* (Right).

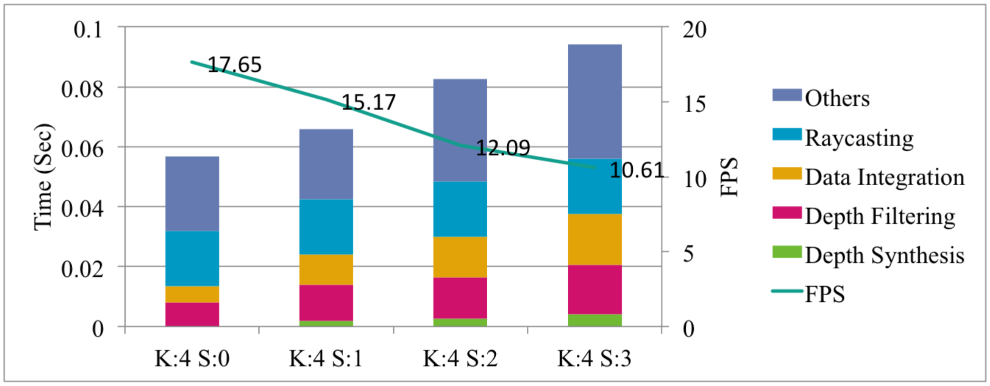


Figure 6.14: Frame rates and cumulative execution times for each process as a function of the number of synthesized depth maps S.

6.4.3 Computational Analysis

The detailed run-time analysis for different number of synthesized views S is given in Figure 6.14. The performance speed of our method is on average 15.17 fps when $S = 1$. GPU programming is used for depth synthesis, depth filtering, data integration, and ray-casting processes. For this test, the size of global volume is $512 \times 512 \times 512$ and the lenght of each side of the voxel is $3.91mm$. To measure time, we perform the overall process 10 times and compute the average.

6.5 Summary

In this work, an accurate real-time dynamic scene reconstruction method using multiple affordable depth cameras is presented. Depth synthesis followed by bilateral filtering allowed us to densify the point cloud from a small set of depth data and eliminate outliers and noise in the raw Kinect data. The proposed weighted TSDF integration attenuates the artifacts of the conventional volumetric representation.

6.6 Acknowledgments

This chapter is in part a reprint of the following publication: Kyoung-Rok Lee, Haleh Azartash, Truong Q. Nguyen, ”Accurate, Real-time, and Dense Surface Recon-

struction of Dynamic Objects using Multiple RGB-D Cameras", *Machine Vision and Applications*, (submitted), 2014.

Chapter 7

Conclusion and future work

This dissertation presents novel efficient and robust algorithms which produce accurate reconstructions of 3D scenes. In particular, we propose a novel depth map enhancement method, robust camera tracking, realistic stationary object reconstruction using a handheld RGB-D camera, and dynamic scene reconstruction using multiple RGB-D cameras. In this Chapter, we review our contributions and present future work.

7.1 Review of Contributions

The contributions of this dissertation are summarized as follows

Depth refinement method Despite depth maps are an important integral component of 3D reconstruction processing, they are often captured at low quality or low resolution due to sensor hardware limitations. In Chapter 3, we have presented a new depth map refinement method which enhance noisy or low-resolution depth maps using high-resolution color images. Every pixel in the region of low reliability collects samples from the region of high reliability and selects the best sample with the highest fidelity. Then, each pixel's selected depth sample is refined by sharing its information with its neighbors' selected samples in the sample refinement stage. Finally, a robust multi-lateral filter, is applied to reduce noise while preserving sharpness along edges. With sample selection and refinement in conjunction with multi-lateral filtering, the proposed method significantly outperforms other state-of-the-art depth refinement methods.

Robust camera tracking method In Chapter 4, we have presented a robust camera estimation method which is effective in challenging situations such as during fast camera motion or in geometrically featureless scenes. A robust orientation estimation based on quaternion method for initial sparse estimation is presented, which requires no prior or small movement assumption. In addition, a weighted ICP (Iterative Closest Point) method for better rate of convergence in optimization and accuracy in resulting trajectory is proposed, which achieves robustness by emphasizing the influence of points that contain more geometric information of the scene. We showed quantitative results on an RGB-D benchmark dataset and demonstated that our method estimates more accurate camera trajectories than other state-of-the-art camera pose estimation methods.

3D reconstruction of static objects In Chapter 6, we have presented a 3D reconstruction method of static object using a handheld RGB-D camera. In our approach, high quality RGB images is acquired from an HD RGB camera which is attached onto a Kinect to reconstruct a 3D model with realistic surface geometry and high-quality color textures. We extended the sampling-based robust multilateral filter presented in Chapter 3 and utilized high frequency information in color images to estimate an objects geometry. Besides, the weighted ICP method presented in Chapter 4 is used to estimate the orientation of the camera in the global coordinate system is estimated for registration between frames. Finally, the registered data is fused together to update a single global 3D volume and visualized as either a rendered image or a mesh model.

3D reconstruction of dynamic objects In Chapter 6, we have presented a 3D reconstruction method of dynamic objects using multiple RGB-D camera. For accurate dynamic scene reconstruction, we presented a depth synthesis technique that generates synthesized depth maps at virtual viewpoints to densify the point cloud so that it covers some surfaces that may be inaccessible to the cameras. In addition, we introduced a reliability-based weight function for integration process to attenuate the artifacts of unreliable depth measurements. We compared our results with existing methods by calculating distance error against laser scanned models and showed accuracy of our proposed method by comparing reconstructed mesh model results with biometric ground-truth.

7.2 Future Work

Future research directions in 3D reconstruction of static and dynamic objects using RGB-D images are:

- Recently, Microsoft released the second version of Kinect. With higher depth precision and a remarkably reduced noise floor, the new Kinect produces better geometry information of the scene and detects smaller objects that were not detected with the old Kinect. We intend to further improve the proposed reconstruction system by adopting the improved RGB-D sensor.

- A drawback of volumetric representation method in our approach is that the system requires large GPU memory for large-scale scene. For building a global model, we use a GPU memory for parallel computation. With a fixed size of GPU memory, the size of global volume is limited, e.g., $512 \times 512 \times 512$. Therefore, a scene with farther objects to capture yields lower volume resolution in the algorithm and consequently degrades the accuracy of the resulting reconstruction. In future work, we plan to address this issue by using an adaptive and sparse grid representation instead of using the dense volumetric representation.

- We plan to apply our dynamic scene reconstruction method to home-based physical therapy system. For example, in hospital, physical therapist performs therapy to patient using our system. By capturing motion and 3D models of patient and physical therapist, our method generates an accurate space-time trajectory of the patient and physical therapist interaction. The recorded data can be used for physical therapy at home. The caregiver can perform the same procedure at home with visual feedback such as arrows, arm shadows, and color coding to match the patient's body position with an avatar on a screen.

Bibliography

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR ’11, (Washington, DC, USA), pp. 127–136, IEEE Computer Society, 2011.

[2] A. Geiger, J. Ziegler, and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time.,” in *Intelligent Vehicles Symposium*, pp. 963–968, IEEE, 2011.

[3] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt, “3d shape scanning with a time-of-flight camera,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1173 –1180, june 2010.

[4] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, “Real-time 3d model acquisition,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’02, (New York, NY, USA), pp. 438–446, ACM, 2002.

[5] M. Pollefeys, *Self calibration and metric 3d reconstruction from uncalibrated image sequences*. PhD thesis, Leuven, 1999.

[6] R. Mohr, L. Quan, and F. Veillon, “Relative 3d reconstruction using multiple uncalibrated images,” pp. 543–548, 1995.

[7] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, “Depth mapping using projected patterns,” October 2008.

[8] T. Oggier, “An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger),” *Proceedings of SPIE*, vol. 5249, no. 65, pp. 534–545, 2004.

[9] H. Hirschmuller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *Computer Vision and Pattern Recognition, 2007. CVPR ’07. IEEE Conference on*, pp. 1 –8, june 2007.

[10] J. Diebel and S. Thrun, “An application of markov random fields to range sensing,” in *In NIPS*, pp. 291–298, MIT Press, 2005.

[11] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," in *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, (New York, NY, USA), ACM, 2007.

[12] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, pp. 839 –846, jan 1998.

[13] F. Li, J. Yu, and J. Chai, "A hybrid camera for motion deblurring and depth map super-resolution," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1 –8, june 2008.

[14] D. Chan, H. Buisman, C. Theobalt, and S. Thrun, "A Noise-Aware Filter for Real-Time Depth Upsampling," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*, (Marseille, France), Andrea Cavallaro and Hamid Aghajan, 2008.

[15] O. Gangwal and R.-P. Berretty, "Depth map post-processing for 3d-tv," in *Consumer Electronics, 2009. ICCE '09. Digest of Technical Papers International Conference on*, pp. 1 –2, jan. 2009.

[16] A. K. Riemens, O. P. Gangwal, B. Barenbrug, and R. P. M. Berretty, "Multistep joint bilateral depth upsampling," *Proceedings of SPIE*, vol. 7257, pp. 72570M–72570M–12, 2009.

[17] F. Garcia, B. Mirbach, B. Ottersten, F. Grandidier, and A. Cuesta, "Pixel weighted average strategy for depth sensor data fusion," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 2805 –2808, sept. 2010.

[18] D. Yeo, E. ul Haq, J. Kim, M. Baig, and H. Shin, "Adaptive bilateral filtering for noise removal in depth upsampling," in *SoC Design Conference (ISOCC), 2010 International*, pp. 36 –39, nov. 2010.

[19] F. Garcia, D. Aouada, B. Mirbach, T. Solignac, and B. Ottersten, "A new multi-lateral filter for real-time depth enhancement," in *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pp. 42 –47, 30 2011-sept. 2 2011.

[20] Q.-Q. Yang, L.-H. Wang, D.-X. Li, and M. Zhang, "Hierarchical joint bilateral filtering for depth post-processing," in *Image and Graphics (ICIG), 2011 Sixth International Conference on*, pp. 129 –134, aug. 2011.

[21] F. Garcia, D. Aouada, B. Mirbach, T. Solignac, and B. Ottersten, "Real-time hybrid tof multi-camera rig fusion system for depth map enhancement," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 1 –8, june 2011.

[22] Q. Yang, R. Yang, J. Davis, and D. Nister, "Spatial-depth super resolution for range images," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1 –8, june 2007.

[23] E. S. L. Gastal and M. M. Oliveira, "Shared sampling for real-time alpha matting," *Computer Graphics Forum*, vol. 29, pp. 575–584, May 2010. Proceedings of Eurographics.

[24] Z. W., A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600 –612, april 2004.

[25] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *Robotics Automation Magazine, IEEE*, vol. 18, no. 4, pp. 80–92, 2011.

[26] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, (Nara, Japan), November 2007.

[27] F. Steinbruecker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[28] T. Tykkala, C. Audras, and A. Comport, "Direct iterative closest point for real-time visual odometry," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 2050–2056, 2011.

[29] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments," in *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[30] I. Dryanovski, R. G. Valenti, and J. X., "Fast visual odometry and mapping from rgb-d data," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2305–2310, 2013.

[31] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, (Vasteras, Sweden), April 2011.

[32] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (St. Paul, MA, USA), May 2012.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[34] O. D. Faugeras and M. Hebert, “The representation, recognition, and locating of 3-d objects,” *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 27–52, 1986.

[35] Z. Zhang, Z. Zhang, P. Robotique, and P. Robotvis, “Parameter estimation techniques: A tutorial with application to conic fitting,” *Image and Vision Computing*, vol. 15, pp. 59–76, 1997.

[36] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’96, (New York, NY, USA), pp. 303–312, ACM, 1996.

[37] C. Wu, “Siftgpu: A gpu implementation of scale invariant feature transform,” 2007.

[38] C. Kerl, “Odometry from rgb-d cameras for autonomous quadrocopters,” Master’s thesis, Technical University Munich, Germany, Nov. 2012.

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

[40] K.-L. Low, “Linear least-squares optimization for point-to-plane icp surface registration,” Tech. Rep. TR04-004, Department of Computer Science, University of North Carolina at Chapel Hill, February 2004.

[41] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, “Scanning 3d full human bodies using kinects,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 643–650, 2012.

[42] D. Tenedorio, M. Fecho, J. Schwartzhaupt, R. Pardridge, J. Lue, and J. P. Schulze, “Capturing geometry in real-time using a tracked microsoft kinect,” pp. 82890A–82890A–14, 2012.

[43] D. Herrera C., J. Kannala, and J. Heikkil, “Joint depth and color camera calibration with distortion correction,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 2058 –2064, oct. 2012.

[44] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, “High-quality single-shot capture of facial geometry,” *ACM Trans. Graph.*, vol. 29, pp. 40:1–40:9, July 2010.

[45] P. Besl and H. McKay, “A method for registration of 3-d shapes,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 239 –256, feb 1992.

[46] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image Vision Comput.*, vol. 10, pp. 145–155, Apr. 1992.

[47] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *INTERNATIONAL CONFERENCE ON 3-D DIGITAL IMAGING AND MODELING*, 2001.

[48] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer, "A Survey of Methods for Volumetric Scene Reconstruction from Photographs," pp. 81–100.

[49] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, pp. 29:1–29:13, July 2013.

[50] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 519–528, June 2006.

[51] J.-M. Hasenfratz, M. Lapierre, and F. Sillion, "A real-time system for full body interaction with virtual worlds," in *Proceedings of the Tenth Eurographics Conference on Virtual Environments*, EGVE'04, (Aire-la-Ville, Switzerland, Switzerland), pp. 147–156, Eurographics Association, 2004.

[52] R. Vasudevan, G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Bajcsy, and K. Nahrstedt, "High-quality visualization for geographically distributed 3-d teleimmersive applications," *Multimedia, IEEE Transactions on*, vol. 13, pp. 573–584, June 2011.

[53] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg, "Omnikinect: Real-time dense volumetric data acquisition and applications," in *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*, VRST '12, (New York, NY, USA), pp. 25–32, ACM, 2012.

[54] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras," *Multimedia, IEEE Transactions on*, vol. 15, pp. 339–358, Feb 2013.

[55] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3d reconstruction and texture mapping from multiple kinects," in *IVMSP Workshop, 2013 IEEE 11th*, pp. 1–4, June 2013.

[56] A. Maimone, J. Bidwell, K. Peng, and H. Fuchs, "Enhanced personal autostereoscopic telepresence system using commodity depth cameras," *Computers & Graphics*, vol. 36, no. 7, pp. 791 – 807, 2012. Augmented Reality Computer Graphics in China.

[57] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim, "Shake'n'sense: Reducing interference for overlapping structured light depth cameras," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 1933–1936, ACM, 2012.

[58] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pp. 51–54, March 2012.

[59] L. Guan and M. Pollefeys, "A Unified Approach to Calibrate a Network of Camcorders and ToF cameras," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*, (Marseille, France), Andrea Cavallaro and Hamid Aghajan, 2008.

[60] C. Witzgall, G. Cheok, and A. Kearsley, "Recovering spheres from 3d point data," in *Applied Imagery and Pattern Recognition Workshop, 2006. AIPR 2006. 35th IEEE*, pp. 8–8, Oct 2006.

[61] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. B. McDonald, "Robust tracking for real-time dense RGB-D mapping with Kintinuous," Tech. Rep. MIT-CSAIL-TR-2012-031, Computer Science and Artificial Intelligence Laboratory, MIT, Sept. 2012.

[62] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, vol. 19, pp. 245–254, July 1985.

[63] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 163–169, Aug. 1987.

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