2013

A ROBUST RGB-D SLAM SYSTEM FOR 3D ENVIRONMENT WITH PLANAR SURFACES

Po-Chang Su
University of Kentucky, iamsue110@hotmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Po-Chang Su, Student

Dr. Sen-Ching S. Cheung, Major Professor

Dr. Zhi D. Chen, Director of Graduate Studies

A ROBUST RGB-D SLAM SYSTEM FOR 3D ENVIRONMENT WITH PLANAR SURFACES

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Electrical Engineering
in the College of Engineering
at the University of Kentucky

By

Po-Chang Su

Lexington, Kentucky

Director: Dr. Sen-Ching S. Cheung, Professor of Electrical and Computer

Engineering

Lexington, Kentucky

2013

ABSTRACT OF THESIS

A ROBUST RGB-D SLAM SYSTEM FOR 3D ENVIRONMENT WITH PLANAR SURFACES

Simultaneous localization and mapping is the technique to construct a 3D map of unknown environment. With the increasing popularity of RGB-depth (RGB-D) sensors such as the Microsoft Kinect, there have been much research on capturing and reconstructing 3D environments using a movable RGB-D sensor. The key process behind these kinds of simultaneous location and mapping (SLAM) systems is the iterative closest point or ICP algorithm, which is an iterative algorithm that can estimate the rigid movement of the camera based on the captured 3D point clouds. While ICP is a well-studied algorithm, it is problematic when it is used in scanning large planar regions such as wall surfaces in a room. The lack of depth variations on planar surfaces makes the global alignment an ill-conditioned problem. In this thesis, we present a novel approach for registering 3D point clouds by combining both color and depth information. Instead of directly searching for point correspondences among 3D data, the proposed method first extracts features from the RGB images, and then back-projects the features to the 3D space to identify more reliable correspondences. These color correspondences form the initial input to the ICP procedure which then proceeds to refine the alignment. Experimental results show that our proposed approach can achieve better accuracy than existing SLAMs in reconstructing indoor environments with large planar surfaces.

KEYWORDS: 3D Reconstruction, Truncated Signed Distance Function, Ray casting TSDF, Iterative Closest Point, Large-scale planar surface alignment

Author's signature: Po-Chang Su

Date: March 13, 2013

A ROBUST RGB-D SLAM SYSTEM FOR 3D ENVIRONMENT WITH PLANAR SURFACES

By

Po-Chang Su

Director of Thesis: Sen-Ching S. Cheung

Director of Graduate Studies: Zhi D. Chen

Date: March 13, 2013

This work is dedicated to my family, especially for my grandfather

ACKNOWLEDGMENTS

First I would like to thank my advisor, Dr. Sen-Ching Samson Cheung, who has guided me research at the Multimedia Information Analysis lab. I had a wonderful learning experience and enjoy a lot of fun in my thesis research. Next, I would like to thank to my thesis committee members, Dr. Laurence Hassebrook and Dr. Ruigang Yang, for their time and valuable suggestions.

I would also like to thank my friends and all the MIA lab members who have supported me and helped me collecting the data during my thesis research, especially Ju Shen and Changpeng Ti. They were always willing to help me when I encounter difficulties.

Finally, I would like to thank my family for their unconditional support and encouragement. Without them, I would never been able to finish my thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we introduce the SLAM techniques, its application and our contributions in reconstructing 3D environments. In the first section, we introduce how SLAM techniques simultaneously track camera's position and generate 3D point clouds. Next, we introduce the SLAM techniques applying in the *virtual mirror rendering system*. In the third and fourth section, we discuss the existing problems in reconstructing 3D environments and provide the solution accurately aligning 3D point clouds. The last section presents the organization of the thesis.

1.1 SLAM Techniques

Simultaneous Localization and Mapping (SLAM) is a technique that uses a mobile camera to reconstruct an unknown 3D environment. Recent works such as [1] focus on using structured-light RGB-D cameras like the Microsoft Kinect to capture both the color and depth data by moving the RGB-D camera by hand through a large environment. Depth images captured by the moving camera are first projected onto a moving 3D coordinate system or a camera pose to create a cloud of 3D points. Based on the similarities between 3D point clouds captured at consecutive time instances, a rigid transformation is then estimated between the two camera poses. Such a process is performed over the entire sequence and a globally-consistent alignment of all point clouds can be obtained by repeated applications of the sequence of estimated rigid

transformations. The global alignment is crucial for the final step of aggregating all the point cloud data into a volumetric representation for noise removal and rendering. The most commonly-used approach to estimate rigid transformations of camera poses from 3D point clouds is the iterated closest point or ICP algorithm [2–7].

1.2 SLAM Applications in the virtual mirror system

As we mentioned in Section 1.1, the indoor environments can be reconstructed by a hand-held RGB-D camera. Our lab's project "Virtual Mirror Rendering System" developed by Dr. Sen-Ching Samson Cheung and Ju Shen uses the reconstructed 3D environments as virtual mirror's background to reduce rendering time [8]. The scanned background can be the large-scale virtual environments which make the virtual mirror system more robust. Virtual mirror rendering is used to simulate a virtual mirror on a computer display. Realistic simulation of a mirror requires viewpoint tracking and rendering, wide-angle viewing of the environment, as well as real-time performance to provide immediate visual feedback. The depth information provided by the RGB-D cameras can be used to track the viewpoint and render the scene from different prospectives. The wide viewing angle of the mirror system is realized by combining the dynamic scene captured by the static camera with a 3D background model created off-line using a color-depth sequence captured by a movable RGB-D camera. Figure 1.1 shows the scanned 3D static background. Figure 1.2 shows the results of merging the pre-scanned 3D static background in the virtual mirror rendering.

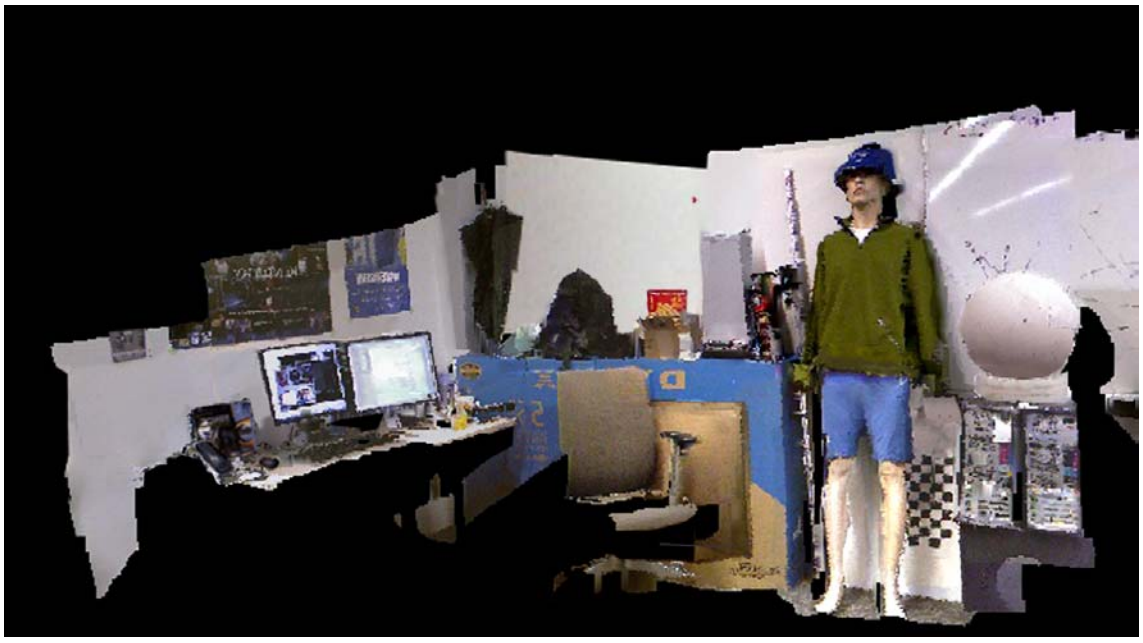


Figure 1.1: Scanned background

Figure 1.2: Mirror View Results: the first row contains the original RGB frames; the second row is the generated mirror view with virtual background; Images from the same column correspond to the same frame in the captured video sequence.

Figure 1.3: Misalignment of a planar surface

1.3 Point Clouds alignment on Planar Surfaces

The alignment accuracy of ICP significantly depends on the scene content. Figure 1.3 shows a virtual view of a vertical wall rendered from a 3D structure created by applying the ICP algorithm from [1] to align 50 frames of moving depth images. One can clearly see that the scene points are grossly misaligned.

The misalignment is caused by the failure of ICP in identifying correct correspondences between planar point clouds of successive frames. Such misalignment error accumulates over multiple frames, making it impossible to process a longer sequence. This is a significant shortcoming of ICP as vertical walls are common in indoor environments.

1.4 Contribution of the Thesis

In this thesis, we propose a novel approach that can accurately reconstruct 3D indoor environments with large planar surfaces using both color and depth features. Our SLAM pipeline is based on that from [1]. An important difference is the use

of color feature descriptors in improving depth data correspondences. Color feature descriptors are first identified from the color images and their correspondences across different frames are robustly identified. These correspondences are then projected onto the 3D coordinate system where they undergo a second stage of noise removal. An initial camera pose transformation is finally estimated which serves as the starting point of the iterative ICP process on the depth data. Our contribution is the development of this new joint color-depth alignment algorithm which produces significant better alignment than those from [1] as demonstrated by our experimental results.

1.5 Organization

The rest of this thesis is organized as follows: In Chapter 2 we review existing techniques for SLAM systems. Chapter 3 first introduces the SLAM system in [1]. We then describe 3D point clouds estimation by Truncated Signed Distance Function (TSDF) voxel structure and the virtual camera ray casting TSDF structure. In Chapter 4, we propose our novel point cloud registration approach that reconstructs 3D environments with large planar surface. Experimental results are shown in Chapter 5, followed by conclusions and future work in Chapter 6.

Chapter 2

Related Work

In this chapter we review existing techniques for SLAM systems and analyze various point clouds registration and pose optimization approaches in the SLAM systems. In the first section we discuss the SLAM systems by using a conventional hand-held camera. The applications include augmented reality, image mosaicing, sparse feature points and dense scenes reconstruction. In the second section we discuss the SLAM systems by using a commodity RGB-D camera reconstructing 3D scenes. In the third section we discuss using portable backpack SLAM system scanning and reconstructing 3D indoor environments. The backpack is equipped with various types of sensors to handle different scenarios.

2.1 SLAM With a Conventional Camera

SLAM techniques have been used for twenty years. The early research on SLAM is usually on the mobile robot, equipped with laser range-finders for navigation. The significant breakthroughs in this field have come from using a hand-held freely moving camera in real time performance [9] [10]. The authors proposed a probabilistic SLAM system that continually corrects current camera localization based on a growing history of past poses. In this system, the key features between frames are selected for camera tracking, and the redundant frames can be removed. Once the camera localization is determined, the sparse sets of features are mapped onto the 3D space.

The camera pose's optimization is performed via probabilistic Bayesian framework, such that tracking errors can be minimized without long-term drift concerns.

The SLAM systems are also applied on augmented reality (AR) research. In [11], thousands of corresponding feature points are determined in order to build a platform for AR space in real world using camera tracking. While the camera is moving, the AR space is generated, and the virtual characters are put on the virtual platform and interacts with each other. By parallel process tracking and mapping, the system quickly builds a static AR space while the camera scans surrounding environment. It is also working well in the case when the camera is at high speed train building AR space based on the outside environments. However, if the camera tracking relies on object corners' features, it occurs tracking failures when rapid camera motion produce blurry images at the object's corners.

SLAM is also commonly used in mosaicing sequential 2D images. The first drift-free, consistent spherical mosaicing images in real time is built in [12]. Extended Kalman Filter (EKF) with SLAM provides a stable way of mosaicing images with 360 degrees pan. Unlike conventional approaches of relying on local feature correspondences to align neighboring images, the authors instead used the whole image sets of feature points to estimate and refine the camera pose. The feature point sets are formed into triangles among neighboring feature points as the skeleton of the global map. With the new image integrating with the current spherical mosaicing images, the global map is updated and warped into the proper appearance. The system is further improved in [13]. Inspired by [11], parallel implementation is utilized to improve performance. One thread is responsible for tracking camera motion optimized by effi-

cient second-order minimization (ESM). Specifically, the authors used only first-order term of ESM cost function and proved that the convergence in several iterations is more stable than the Lucas-Kanade method. Another thread is responsible for global optimization and consistent map based on key frames information.

In [14], dense 3D scenes are rapidly reconstructed with a single live camera. For the system input, camera poses are estimated by structure from motion (SFM) and used to generate sparse point clouds. These points form implicit surface initially. The key frames are selected from the local camera pose and used to refine the local surface. A global consistent dense 3D scenes is obtained through combing local reconstructions together. The camera pose estimation is further improved in [15]. Accurately local depth maps are generated by solving the cost function from hundreds of neighboring images' photometric information. The system also have ability of occlusion handling. However, the system operate normally only under the circumstance that the illumination stay consistency. The system will suffer from tracking failure when the illumination suddenly changes.

2.2 SLAM With a Commodity RGB-D Camera

Traditional 3D scenes reconstructions are created by structure from motion (SFM) or multi-view stereo (MVS). Recently, with the commodity RGB-D camera releasing, depth map obtained from the RGB-D camera can easily convert the 2D pixels into the 3D points. Henry et.al [16] [17] are the pioneers in proposing reconstructing 3D dense scenes by using a hand-held commodity RGB-D camera. The camera pose estimation is based on the color information combining with geometry data after several

iterations of ICP. Next, pose optimization TORO [18] and dense scenes modeling SURFELS [19] are used to achieve globally consistent map. The authors showed that their reconstructed results are well-aligned.

To enable the SLAM system operate in real-time and have intuitive interaction interface with non-expert users, Du in [20] proposed improving SLAM system to let non-expert users creating their own 3D scenes reconstruction. The users can obtain instant feedback while scanning indoor scenes. The capability of recovering camera's trajectory from the failure tracking part help users building a large-scale 3D map and maintain the 3D map globally consistent.

In [21] [22], the authors proposed a novel approach in reconstructing 3D scenes. First, feature descriptors are extracted from the color images by SURF [23] and converted into global 3D points according to the camera localization. RANSAC [24] is used to estimate the transformation among feature point sets. The pose optimization is performed to refine the camera localization. This approach is similar to ours. However, they use 3D points instead of voxels in the ICP procedure, which is prone to drifting error. Our method uses a TSDF structure which enables well-aligned edges.

Instead of extracting feature points matching two frames, the authors in [25] [26] defined a warping function that warp current image's gray level and depth information to the reference image. Cost function is minimized between the warped image and the current image. A M-estimator is used to estimate robust camera pose and eliminate outliers and dynamic objects. The key frames selection and multi-resolution approach make the system run in real-time.

2.3 Portable Backpack SLAM System

Although the SLAM systems on mobile robots are well-developed area, there exist the limitations on Geographical situations. For example, conducting experiments in the stairwell make the robots have difficulties moving up and down stairs. Therefore, the human-operated backpack SLAM system were proposed [27] and had more flexibility on scanning indoor environments, even scanning on the different floors in the same building. The backpack SLAM system is composed of four 2D scanners and two inertial measurement units (IMU's). Each scanner scans a 2D plane and is orthogonal to each other, while the IMU's not only provide the orientation estimation, but also refine 6 DOF transformation. The authors switched different point clouds registration approaches to handle different types of scenes.

The authors in [28] proposed improving backpack SLAM system that has the capability scanning more complex environments such as T-shaped corridor intersection and two different indoor hallways connected by staircases. The automatic loop detection based on [29] [30] produces accurate camera localization in scanning complex scenes. To render the global map without misalignment between successive frames, the backpack is added a camera providing imagery, and image based pose estimation algorithm [29] [24] refines the results from scanning matching based algorithm.

In [31], the backpack is added two cameras on the left and right to further reduce localization errors. The system takes the advantage of planar scenes data under the assumption that the camera localization in planar scenes is accurate. Like previous backpack systems [27], the adaptive localization algorithm is used to handle the dif-

ferent scenarios such as complex scenes or planar scenes, and the portions of planar scenes data are highly trusted and used in ICP. In the case of scanning staircases, the camera which faces planar walls determines camera localization. Either of camera is responsible for backpack's localization when traversing up and down staircases. Localization errors are minimized by loop closure events obtained by these two cameras.

Chapter 3

3D Point Clouds Estimation and Representation

In this chapter we discuss the approaches used in our SLAM system in estimating 3D points and rendering virtual views. In Section 3.1, we provide an overview of our SLAM system. We then introduce the essential vision concept, which will be used in the later discussion. In Section 3.2, we discuss merging all the depth data into a voxel-based structure. Each voxel has a signed distance function showing how far each voxel is from the nearest surface point. In Section 3.3, we discuss estimating surface points by creating a virtual camera ray casting the 3D grid structure. The surface point is generated while the ray passes through the zero-crossing region in the 3D gird structure. In Section 3.4, we discuss virtual view rendering. The estimated surface points are mapped with color values from RGB images and applied a layered interpolation to fill in the gaps on the rendered image.

3.1 Introduction

3.1.1 System Overview

Our SLAM system is based on [1]. A volumetric 3D grid structure called Truncated Signed Distance Function (TSDF) is used to aggregate 3D data obtained by depth images. Each voxel is signified by a signed distance value from the nearest 3D point. Using the reconstructed TSDF structure, virtual camera views can be rendered

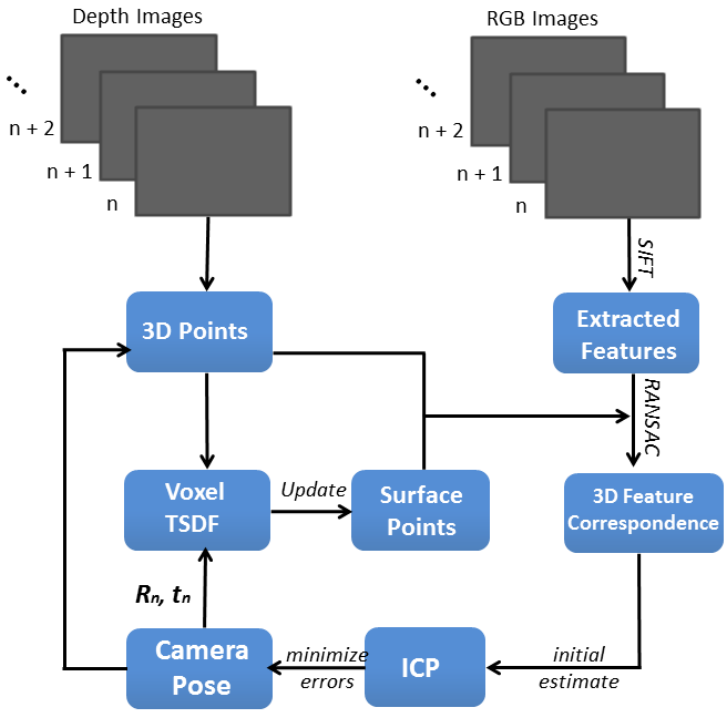


Figure 3.1: Overview of the SLAM system

from an arbitrary camera pose – the virtual camera can ray-cast the TSDF structure to identify the estimative surface points. Specifically, the zero-crossing region is identified via a fast search procedure and the estimative surface points are interpolated within the zero-crossing region. During the model construction stage, ICP is used to estimate the camera pose with respect to the global coordinate system adopted by the TSDF structure. Instead of using the typical frame-to-frame tracking method which is prone to drifting error, the depth data are denoised and then aligned against the estimative surface points produced from the interpolated TSDF structure. Missing depth information is incrementally filled with depth data captured at different frames. Figure 3.1 is the overview of the SLAM system.

3.1.2 Calculate 3D points from depth maps

Before we send the data into the SLAM system, we first convert depth maps obtained from the RGB-D camera into 3D points. The intuitive way is to calculate the similar triangular method to obtain the 3D corresponding point (X, Y, Z) from the 2D pixel (U, W), which is denoted by p. We first calculate the corresponding X value in the 3D coordinate. The horizontal distance from U to the image central C_x over the focal length f_x is proportional to the horizontal distance from the pixel's corresponding 3D value X to the central C_X over the depth Z.

$$\frac{|U - C_x|}{f_x} = \frac{|X - C_X|}{Z} \tag{3.1}$$

The Equation (3.1) can be rewritten as

$$X = C_X + \frac{Z(|U - C_x|)}{f_x} \tag{3.2}$$

the coordinate Y can be computed in the same way:

$$Y = C_Y + \frac{Z(|W - C_y|)}{f_y} \tag{3.3}$$

Equation 3.2 and 3.3 can be rewritten as

$$u(U, W) = ZK^{-1} \begin{bmatrix} U \\ W \\ 1 \end{bmatrix} \tag{3.4}$$

Where u maps the 2D image pixel (U, W) to its 3D coordinate, and K is the camera intrinsic matrix

$$K = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Equation 3.4 can be simplified as $u = K^{-1}\dot{p}$. We use a dot notation to represent homogeneous vectors $\dot{p} = (p^\top|1)^\top$. Conversely, K can also be used to project the 3D point onto the 2D point

$$\begin{bmatrix} U \\ W \\ 1 \end{bmatrix} = \frac{1}{Z}\left(K\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}\right) \tag{3.6}$$

The Equation 3.4 can be simplified as $\hat{p} = Ku$. $\hat{p}$ represent projected points from the 3D space.

3.1.3 3D Projections and 6 DOF transformation

In our research, the data are often transformed between 2D and 3D. Homogeneous coordinates provide a bridge to project between the 3D point and the 2D plane. After the 3D point is multiplied by the camera intrinsic matrix, the Z value is scaled to 1 and becomes $(X/Z, Y/Z, 1)$ in the homogeneous coordinate system. The point in the 2D plane is $(X/Z, Y/Z)$. Six degrees of freedom transformation matrix T can convert the 3D point between the camera coordinate system and the global coordinate system. T is controlled by the rotation roll, yaw, pitch, translation X, Y, and Z

$$T = \begin{bmatrix} R_r & T_t \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

Rotation R_r is the 3×3 matrix and translation T_t is the 3×1 matrix. Calculating the 3D point at the n^{th} frame in the global coordinate space is as follows

$$u_n = R_r^n \cdot u + T_t^n \tag{3.8}$$

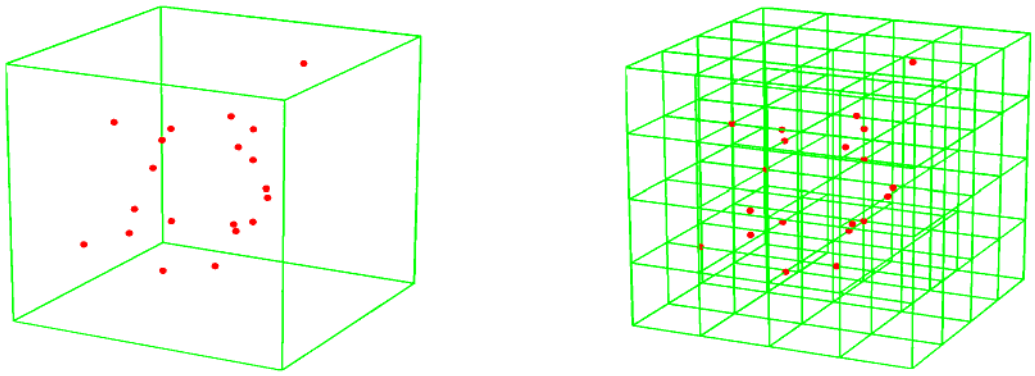
The Equation (3.8) can be written as

$$\dot{u_n} = T \cdot \dot{u} \tag{3.9}$$

Where $\dot{u} = (u^\top|1)^\top$.

3.2 Constructing TSDF structure from the 3D data

The raw depth values obtained from the RGB-D camera are the uncertainty measurements along with noise. Therefore, the 3D point clouds we compute from the raw depth maps may not be aligned accurately between frames. A precise 3D point estimation method is required in order to build the well-aligned and detailed 3D virtual scenes. To accomplish this goal, first, we construct the truncated signed distance functions (TSDF) [32] in the voxel-based structure. The structure is composed of a number of voxels as shown in Figure 3.2. The TSDF voxel structure can be used to aggregate multiple raw depth images into the same coordinate system and to generate interpolated 3D point clouds for rendering. The TSDF value stored at each



(a) (b)

Figure 3.2: TSDF voxel-based structure construction. The cuboid in Figure 3.2a is divided into the voxel-based structure as shown in Figure 3.2b. The red dots are the 3D data obtained from the raw depth maps

voxel is based on the signed distance towards the closest 3D scene point. The sign of this distance value is based on whether the voxel is in front of or behind the implicit

surface as defined by the local surface normal. From the camera's perspective, if the voxel is behind the surface, a negative sign is assigned. If the voxel is in front of the surface, a positive sign is assigned. In general, we use brute-force search or KD-tree search method to search the nearest point. However, it is not the efficient ways to search the nearest point in the dense point clouds. Instead, we adopt the parallel projective TSDF approach [1, 33] to compute each voxel's TSDF. Each voxel V in the global coordinate space is first converted back to the camera coordinate space, and then onto the 2D image plane. We search the nearest pixel $p = (U, W)$ on the image plane. We exclude the projective pixel (U, W) if it is out of the image range: $0 < U \leq 640, 0 < W \leq 480$

$$p = \lfloor (KT_n^{-1}V) \rfloor \tag{3.10}$$

Where K is camera intrinsic matrix, T_n is the 6-DOF transformation at the n^{th} frame. We also compute the ray direction in the global coordinate space at the n^{th} frame

$$\lambda = \|T_n K^{-1} \dot{p}\|_2 \tag{3.11}$$

Each voxel is assigned a signed distance value

$$S_n(V) = (d_n(p) - \lambda^{-1}\|V - t_n\|_2) \tag{3.12}$$

Where $\lambda^{-1}\|V - t_n\|_2$ is the depth value from the camera to the voxel, $d_n(p)$ is the depth value at pixel p in the n^{th} frame, and t_n is the camera location at the n^{th} frame. We then checked whether the voxel is in front of the camera's perspective view while the voxel is projected onto the image plane. If the voxel is behind the camera's perspective view, we don't update the voxel's signed distance value at the n^{th} frame.

The vector P is from the camera to the voxel V. The vector Q is the $l2$ norm of the central pixel's ray direction. The voxel's signed distance function is valid if the dot product of vectors P and Q is grater than zero. Once we verify all the valid voxels which are in front of camera's perspective, the signed distance value is "truncated" as only the voxel values that are sufficiently closed to the surface are recorded. We compute the voxels' truncated signed distance functions S_{T_n} if the voxels are within the truncated signed distance range as follows

$$S_{T_n}(V) = \begin{cases} min(1, \frac{S_n(V)}{\mu}) & \text{iff} \quad \mu \geq S_n(V) > 0 \\ max(-1, -\frac{S_n(V)}{\mu}) & \text{iff} \quad 0 > S_n(V) \geq \nu \\ null & \text{otherwise} \end{cases} \tag{3.13}$$

Where S_{T_n} is the truncated signed distance function at the n^{th} frame, μ is the maximum truncated signed distance, and ν is the minimum truncated signed distance. We then proceed to update the global TSDF S_T using the current frame's S_{T_n}. An exponentially weighted average is applied to the TSDF value at each voxel that is sufficiently close to a 3D point from the current frame

$$S_T(V) = \frac{W_n(V)S_{T_n}(V) + W_{n-1}(V)S_{T_{n-1}}(V)}{W_n(V) + W_{n-1}(V)} \tag{3.14}$$

$$W(V) = W_n(V) + W_{n-1}(V) \tag{3.15}$$

The averaging helps to remove inherent in the depth images data and multiple passes of all the depth images are usually required to produce a consistent and stable TSDF structure. Finally, the global weighted average of truncated signed distance functions are constructed by all the frames without error or drifting concerns.

3.3 Surface points estimation by ray casting TSDF structure

Given a TSDF structure, we can generate the estimative 3D points by first ray casting from the virtual camera and traverse through the voxel structure. We adopt the fast voxel traversal algorithm in [34]. A visible surface point is identified when the ray passes through a zero-crossing region in which the distance values change from positive to negative. The ray stops passing through the voxel structure when the zero-crossing region or the back side of the object is found, or the ray ultimately exits the TSDF voxel structure. Each pixel's corresponding ray direction λ in the global coordinate space is the same as Equation (3.11). The distance from the camera to the surface point is $\lambda d_n(p)$. Truncated values are used to decrease the voxel traversal time. In other words, the ray can start from the truncated distance $\lambda d_n(p) - \mu$ instead of starting from the camera. The μ is the truncated value. The truncated point X_t is

$$X_t = T_n D K^{-1} \dot{p} \tag{3.16}$$

Where D is the depth value from camera to the truncated point

$$D = \frac{\lambda d_n(p) - \mu}{\lambda} \tag{3.17}$$

The Figure 3.3 shows the ray passing through the TSDF structure from the virtual camera and truncated point, respectively. Once the zero-crossing region is identified, linear interpolation is used to estimate a surface point u_s based on the TSDF values of the two boundary voxels. The trilinear points V_a and V_b in the voxels are extracted and used as the linear interpolation's reference points

$$u_s = \begin{cases} V_a + \frac{S_T(V_a)}{S_T(V_b)} \|V_a - V_b\|_2 & \text{iff} \quad \lambda > 0 \\ V_a - \frac{S_T(V_a)}{S_T(V_b)} \|V_a - V_b\|_2 & \text{iff} \quad \lambda < 0 \end{cases} \tag{3.18}$$

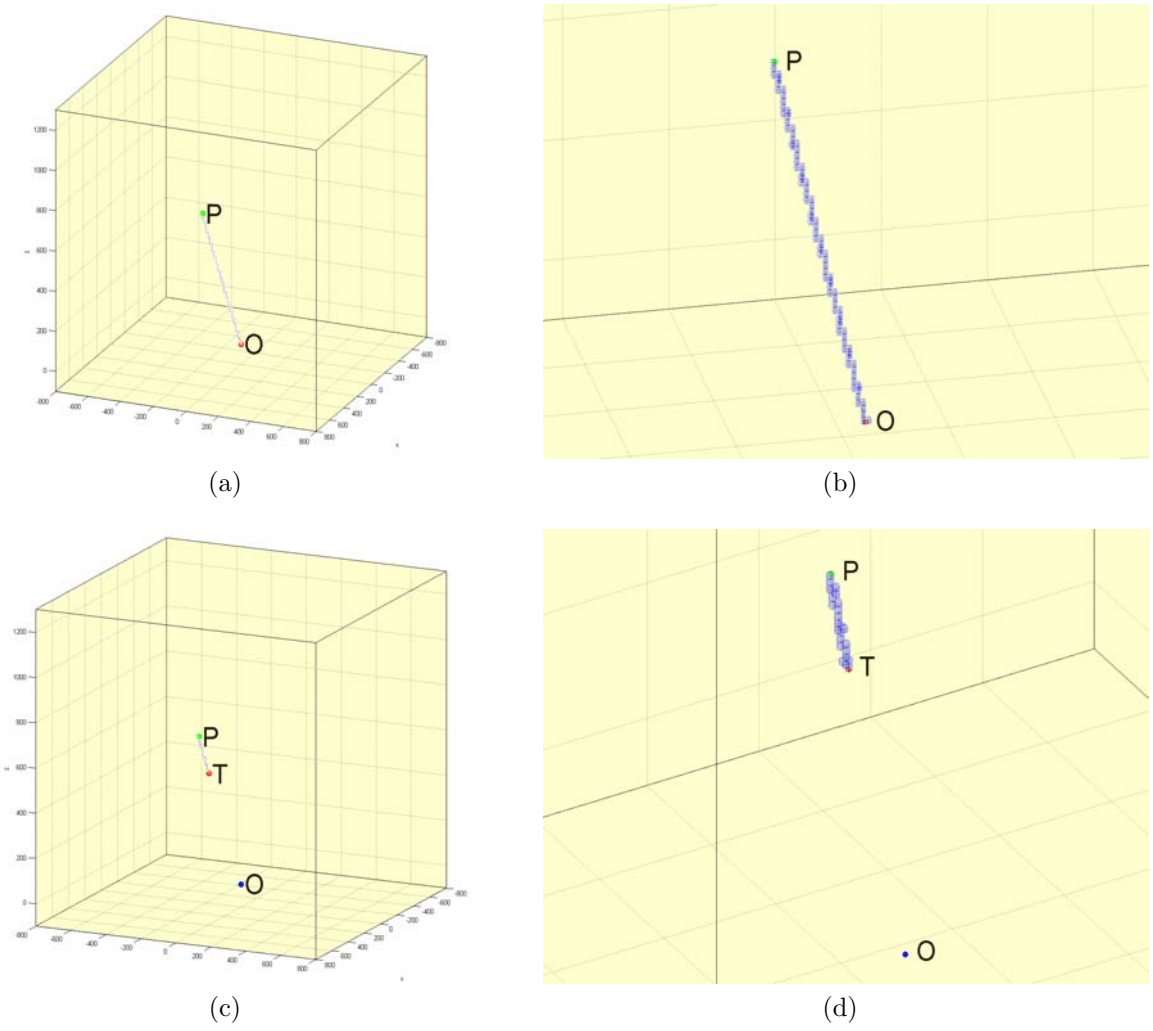


Figure 3.3: Ray passing through the TSDF voxel structure. Figure 3.3a and 3.3b show that the ray passes through the TSDF structure starting from the virtual camera. Figure 3.3c and 3.3d show that the ray passes through the TSDF structure starting from the truncated point by using ray skipping method. The small blue cubes are the paths that the ray passing through. The blue dot O is the virtual camera location, the red dot T is the truncated point, and green dot P is the estimative surface point

Where V_a is in front of the surface and V_b is behind the surface. All the identified surface points from a 3D point clouds as if it was captured from a virtual camera placed at (X, Y, Z). In addition, the higher resolution voxels (dividing the same size of voxel structure into a number of smaller size voxels) we create, the more accurate

surface points we obtain. For the voxel's trilinear point where is very close to the surface, we can assume that this point's normal vector is roughly the same as the surface point's normal vector which has the same ray passing through from the virtual camera. To compute this point's normal vector, we take the derivative of $S_T(V)$

$$N_s = (\frac{\partial S_T(V)}{\partial x}, \frac{\partial S_T(V)}{\partial y}, \frac{\partial S_T(V)}{\partial z})$$

The N_{sur} can be expanded as follows

$$\begin{aligned}
\frac{\partial S_T(V)}{\partial x} &= \frac{S_T(x+1,y,z) - S_T(x-1,y,z)}{2} \\
\frac{\partial S_T(V)}{\partial y} &= \frac{S_T(x,y+1,z) - S_T(x,y-1,z)}{2} \\
\frac{\partial S_T(V)}{\partial z} &= \frac{S_T(x,y,z+1) - S_T(x,y,z-1)}{2}
\end{aligned}$$

Before we do the point clouds registration in the next chapter, we make all the normal vectors pointing outwards the surfaces. We use simple dot product checking method to check whether the normal vectors is pointing outwards the surfaces. Beside the N_s vector we have obtained, the vector G is formed from the surface point u_s to the camera. If the dot product of these two vectors is negative, the N_s is flipped 180 degrees.

3.4 Virtual view rendering

After obtaining the estimative points by raycasting TSDF structure, each estimative point is mapped with a color value based on RGB image. In the raycasting process, each pixel has a corresponding ray passing through TSDF voxel structure, the pixel's color value is directly mapped to the corresponding estimative point. We

use OpenGL to render the estimative points. Since the reconstructed scene is just a group of discretely distributed point cloud. The generated image may have many unrendered regions caused by the gaps between neighboring points. We apply a layered interpolation to fill in the gaps on the rendered image. This process can be described as follows: for the pixels with no display points, they need to be interpolated from neighboring pixels. A naive approach would be to perform spatial interpolation after obtaining the color values for all the pixels that contain at least one display points. We notice that this approach creates a great deal of blending of scene objects at different depth. To better preserve object boundaries, we separate the rendering into two phases based on the depth values from the scene points those that are at or closer than the viewer and those that are beyond. These two sets typically have very different depth values. We first start with the latter group with scene points that are far away, apply the above process of identifying color for each pixel and then perform interpolation on both depth and color values to fill in small gaps. These interpolated values are inserted back to the data structure of the closer pixels as if they are from the true 3D point clouds. In the second phase, we render all these closer pixels, select the correct color value based on both 3D point clouds and interpolated results, and finally perform one more round of interpolation just on the color values. Such a layered approach provides a far sharper object boundaries as it respects the inherent depth values. It is possible to increase the number of depth levels to create a better rendering but two levels are sufficient for our application.

Chapter 4

Point Clouds Registration and Camera Pose Estimation

In Chapter 3, we have obtained the surface points from TSDF structure. In this chapter we describe the SLAM system estimating the camera pose. In Section 4.1, we discuss using Bilateral filter removing noises on raw depth images. We test different Bilateral filter parameters filtering raw depth images and compare the corresponding filtered depth images. In Section 4.2, we discuss estimating camera pose according to the previous estimated surface points. In Section 4.3, we define the problem on aligning planar surfaces. In Section 4.4, we present our approach estimating camera pose based on joint color-depth information.

4.1 Noise removal by Bilateral filter

Raw depth images taken by the RGB-D camera are noisy. Therefore, noise reduction is a crucial step if we want to calculate accurate 3D points from the depth maps. Without reducing the noise, estimating the correct camera pose will be affected, and the drifting problem will occur. In general, Gaussian filter is commonly used to smooth the images. However, it blurs depth discontinuities in the depth maps. Instead, we use Bilateral filter [35] filtering out the noise from the raw depth images. Bilateral filter is controlled by two parameters: spatial weight and depth weight. According to these two weights, Bilateral filter not only smoothes depth images, but also preserves the depth discontinuities. The spatial weight is determined by how

close between the nearby pixel and the central pixel in the filter window. The shorter Euclidean distance between nearby pixels and the central pixel in the filter window, the higher spatial weight the nearby pixels obtain. The depth weight is determined by the similarity between nearby pixels' depth values and the central pixel's depth value in the filter window. The nearby pixel will receive higher weight if the depth difference between the nearby pixel and the central pixel is smaller. Equation 4.1 computes the filtered depth values by using Bilateral filter.

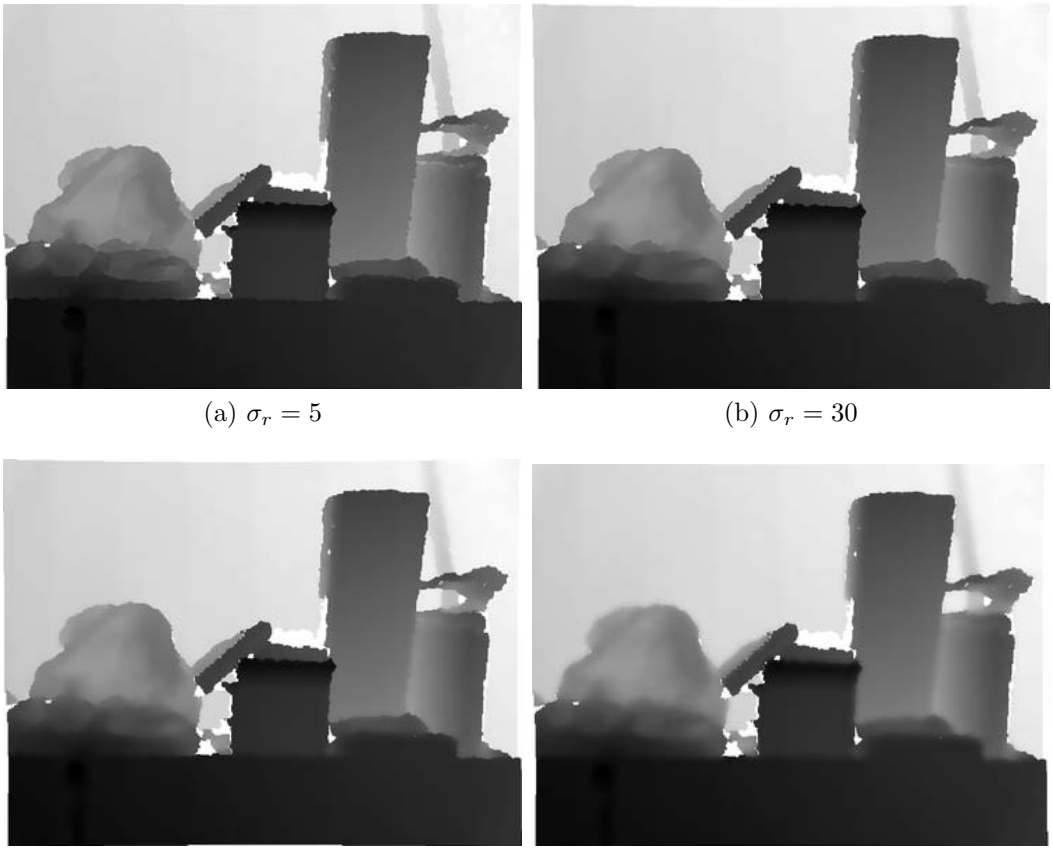
$$D_f(p) = \frac{1}{C_h} \sum_{q \in M} e^{\frac{-\|p-q\|^2}{2\sigma_s^2}} e^{\frac{-|R(p)-R(q)|^2}{2\sigma_r^2}} R(q) \tag{4.1}$$

Bilateral filter window size M is $(2m+1) \times (2m+1)$. Where m is the radius of Bilateral filter window, q are the neighboring pixels surrounding the central pixel p in the filter window, σ_s is the Gaussian spatial parameter, σ_r is the Gaussian depth parameter, and C_h is the normalization constant

$$C_h = \sum_{q \in M} e^{\frac{-\|p-q\|^2}{2\sigma_s^2}} e^{\frac{-|R(p)-R(q)|^2}{2\sigma_r^2}} \tag{4.2}$$

Compared to Gaussian depth parameter σ_r, Gaussian spatial parameter σ_s only has very small influence on the depth maps. In our tests, we fix the Gaussian spatial value and select different Gaussian spatial values to compare the corresponding filtered depth images. In Figure 4.1, from the first row to the second row, we gradually increase the σ_r from 5 to 120. The results shows that the depth image is more smoother when the σ_r gradually increases.

After we calculate the filtered depth value from the depth map, we back-project the filtered depth values to the 3D points in the camera's coordinate space. As we



(a) $\sigma_r = 5$ (b) $\sigma_r = 30$

(c) $\sigma_r = 60$ (d) $\sigma_r = 120$

Figure 4.1: Filtered depth images comparison

mentioned in Section 3.1.2, 3D points are obtained by the following equation

$$u(U, W) = D_f(p)K^{-1} \begin{bmatrix} U \\ W \\ 1 \end{bmatrix} \tag{4.3}$$

Where $D_f(p)$ is the filtered depth value at the pixel (U, W). To compute each 3D vertex's normal vector, we select the neighboring 3D vertices forming the two vectors from the 3D vertex $u(U, W)$ to the neighboring 3D vertices $u(U+1, W)$ and $u(U, W+1)$. The cross product of these two vectors is the normal vector $N(p)$ of the 3D vertex $u(U, W)$.

$$N(p) = (u(U + 1, W) - u(U, W)) \times (u(U, W + 1) - u(U, W)) \tag{4.4}$$

The normal vector's normalization ν

$$\nu = N(p)/\|N(p)\|_2 \tag{4.5}$$

To ensure all normal vectors pointing outwards the surfaces, we use dot product checking normal vector's direction and flip all normal vectors pointing outwards if the normal vectors is pointing inwards originally.

4.2 Camera pose estimation based on the previous surface points

In the RGB-D SLAM system, the camera pose is estimated sequentially by aligning the captured 3D point cloud at the current frame with the predictive surface points obtained from the TSDF structure followed by an Euclidean transformation corresponding to the last estimated camera pose. The alignment is achieved by estimating the rotation and translation based on the iterative Closest Point algorithm (ICP), which minimize the point-pairs between the current frame and the previous frame. The first step of ICP algorithm is to search point correspondences. In this section, we search point correspondences from the previous surface points. Parallel fast projective data association approach in [1] is used to find the point correspondences in the dense point clouds. Since we don't know the new frame's camera pose, we take a initial guess that the new frame's initial camera pose is equal to the previous frame's camera pose

$$u_{new} = T_{n-1}\dot{u}_n(p) \tag{4.6}$$

Now we need to perform ICP for several times, each iteration δ in ICP obtains the relative transformation T_n^δ between iteration δ and $\delta - 1$. Note that T_{n-1} is equal to

$T_n^{\delta=1}$. Equation (4.6) can be rewritten as

$$u_{new} = T_n^{\delta} \dot{u}_n(p) \tag{4.7}$$

We project each new 3D point u_{new} in the current frame onto the previous frame's image plane and search the nearest neighbor to find the corresponding projective pixel

$$\hat{p} = \lfloor (K T_n^{-1} u_{new}) \rfloor \tag{4.8}$$

Where the projective pixel $\hat{p}(U, W)$ must be in the range: $0 < U \leq 640, 0 < W \leq 480$. The projective pixel's 3D point $\hat{u}_{n-1}$ serve as the point correspondence for u_{new}. Point-pairs with unusually large distances or disparate normal directions are excluded for the estimation of the transformation.

$$\| u_{new} - \hat{u}_{n-1}(\hat{p}) \|_2 \leq \phi \tag{4.9}$$

$$(R_n^{\delta} N_n(p), \hat{N}_{n-1}(\hat{p})) \leq \psi \tag{4.10}$$

To speedup the computation time on T_n^{δ}, we assume there is only a small angle change ($\theta \approx 0$) between δ and $\delta - 1$ due to the depth camera capturing frame rate is at 30HZ. Based on this assumption, $sin\theta \approx \theta$ and $cos\theta \approx 1$. The problem is simplified as linear least-square optimization problem [36]. Therefore, The relative transformation T_{inc}^{δ} can be written as

$$T_{inc}^{\delta} = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix} \tag{4.11}$$

The new transformation is equal to relative transformation T_{inc}^{δ} multiplied by previous transformation matrix $T_n^{\delta-1}$

$$T_n^{\delta} = T_{inc}^{\delta} T_n^{\delta-1} \tag{4.12}$$

The updated parameters can be written as a vector

$$\epsilon = (\beta, \gamma, \alpha, t_x, t_y, t_z)^{\top} \tag{4.13}$$

To solve the vector ϵ, we minimize the cost function by computing point pairs' point-plane energy.

$$E_{point-plane} = \sum \|(T_n^{\delta} \dot{u}_n(p) - \hat{u}_{n-1}(\hat{p}))^T \cdot \hat{N}_{n-1}(\hat{p})\|_2 \tag{4.14}$$

The newly estimative 3D position in the global space is $T_n^{\delta}\dot{u}_n(p)$ and can be rewritten as another representation by adding the matrix $g(p)$

$$T_n^{\delta}\dot{u}_n(p) = g(p)\epsilon + u_{new} \tag{4.15}$$

Where $g(p)$ is the skew-symmetric matrix combining with the identity matrix $[G(p)|I]$

$$g(p) = \begin{bmatrix} 1 & -az & ay & 1 & 0 & 0 \\ az & 0 & -ax & 0 & 1 & 0 \\ -ay & ax & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

For each point-pair's cost function, the equation can be written as

$$E_{point-plane} = \hat{N}_{n-1}(\hat{p})^{\top}(g(p)\epsilon + u_{new} - \hat{u}_{n-1}(\hat{p})) \tag{4.17}$$

We take the derivative of cost function and the equation can be rewritten as $A\epsilon = B$

$$\hat{N}_{n-1}(\hat{p})^{\top} g(p)\epsilon = \hat{N}_{n-1}(\hat{p})^{\top}(\hat{u}_{n-1}(\hat{p}) - u_{new}) \tag{4.18}$$

In order to solve the parameter vector ϵ, a 6X6 symmetric linear system is formed by multiplying $A^\top$. The equation can be rewritten as follow

$$\Sigma(A^\top A)\epsilon = \Sigma(A^\top B) \tag{4.19}$$

We use Cholesky decomposition to decompose $A^\top A$. The sum of $A^\top A$ can be decomposed into upper matrix R and lower matrix $R^\top$

$$\Sigma(A^\top A) = R^\top R \tag{4.20}$$

The followings are the equations solving lower matrix $R^\top$ and upper matrix R

$$R^\top y_l = \Sigma(A^\top B) \tag{4.21}$$

$$R\epsilon = y_l \tag{4.22}$$

The above processes are iterated until convergence as the current frame is gradually moved towards the previous frame by the transformation estimated. The rigid transformation is then applied to construct TSDF structure which we have already discussed in Chapter 3.

Estimating camera pose based on the previous surface points is robust when the indoor environments have depth variations. However, it fails on low geometry scenes such as large planar surfaces.

4.3 Problem Statement

To understand the problem on aligning planar surfaces, let us first review the basic procedure of ICP as summarized in Algorithm 1 [2]. Given two consecutive frames of 3D point clouds F_{t-1} and F_t, the algorithm aims at iteratively refining

Algorithm 1 ICP

Require: $F_{t-1} = \{p_1, p_2, \ldots, p_m\} \in \mathbb{R}^3$
$F_t = \{q_1, q_2, \ldots, q_n\} \in \mathbb{R}^3$

1: Initialization:
$s := 0$ and $F^{(s)} := F_t$
2: Identify closest points: $\forall p_i \in F_{t-1}$
$d_i := \min_{q \in F^{(s)}} \|p_i - q\|_2$
$f^{(s)}(p_i) := \begin{cases} \arg\min_{q \in F^{(s)}} \|p_i - q\|_2, & d_i \leq \epsilon \\ \text{unmatched} & \text{otherwise} \end{cases}$
3: Find $R^{(s)}$ and $t^{(s)}$ to minimize the average of
$\|p_i - (R^{(s)} \cdot f^{(s)}(p_i) + t^{(s)})\|_2^2$
among all p_i's with a matching $f^{(s)}(p_i)$
4: Refinement:
$F^{(s+1)} := \{q_i' : q_i' := R^{(R)} \cdot q_i + t^{(s)}\}$
5: $s := s + 1$
6: Go back to step 2 until error in step 3 is below a threshold

a rotation matrix $R^{(s)}$ and a translational vector $t^{(s)}$ which are applied F_t to best align each point in F_{t-1} to its closest point in F_t. The distance parameter ϵ excludes correspondences that are too far apart to be considered as reasonable.

When ICP initially identifies the closest points between the two point clouds, there could be many false correspondences. The goal of ICP is to improve these correspondences by moving two point clouds closer to each other in each iteration. However, the above procedure may fail if the majority of the 3D points fall on a planar surface. This is illustrated in Figure 4.2. The red points from each plane indicate the true correspondences. But the closest-point search wrongly assigns the green points from F_{t-1} to match the points in F_t. If there were significant depth variations among the 3D points, no rigid transformation could produce a good match between these wrong correspondences and step 3 of the ICP algorithm merely produces a transformation that moves the two clouds closer. However, for a planar surface, these

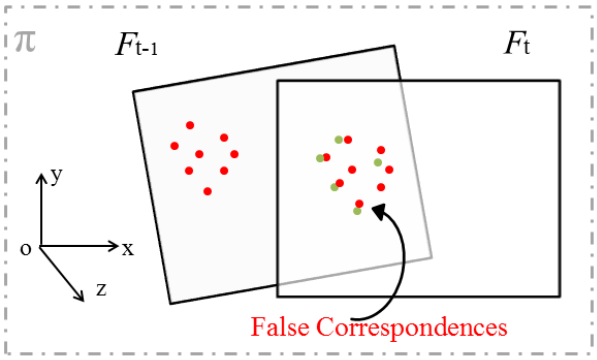


Figure 4.2: How ICP fails to align planar structures

wrong correspondences may lead to a rotation about the x and the y-axis and a translation in the z direction that can completely align the two planes. The lack of depth variations prevents the in-plane rotation and the translation along the $x - y$ plane to be effectively estimated. As such, we have an underdetermined system and the ICP prematurely terminates without providing the true alignment. Notice that such misalignment error accumulates over time and thereby significantly affects the subsequent reconstruction of the 3D structure.

4.4 Joint color-depth camera pose estimation

The proposed camera pose estimation algorithm based on fused color and depth information works as follows. The RGB and depth cameras are extrinsically aligned and temporally synchronized using the OpenNI software library. Our camera pose estimation starts by first extracting the SIFT features from the color frames and search for the closest match between frames [29, 37]. If all the SIFT feature points fall on the same plane, the matching correspondences between frames would be related by a planar homography. However, if the scene structure is more complex, we need a more

robust procedure to identify the subset of correspondences that could fit well within a planar homography. To this end, we use a RANSAC-like procedure to identify such a subset and to estimate the optimal homography between correspondences [24]: all the correspondences are first used to estimate a homography matrix. We then apply the estimated homography to map one set of points to the other and eliminate those pairs that are too far apart. We repeat this process until it converges to a stable subset of correspondences that are well described by a single homography matrix. Finally these corresponding pairs along with the associated depth measurements are projected back onto the 3D space to be used as the initial point correspondences for the ICP algorithm.

Note that the SIFT correspondences are based on the current and previous color frames captured by the camera while the ICP is used to align the current depth frame with the TSDF voxel structure. We use the SIFT correspondences as the initial match between the estimated surface points from the previous frame and the captured point cloud from the current frame. To ensure a robust matching, we again use RANSAC to find the inliers as a subroutine within ICP – outliers are iteratively removed if they do not agree with the estimated transformation [38] until the procedure converges to a stable set of correspondences.

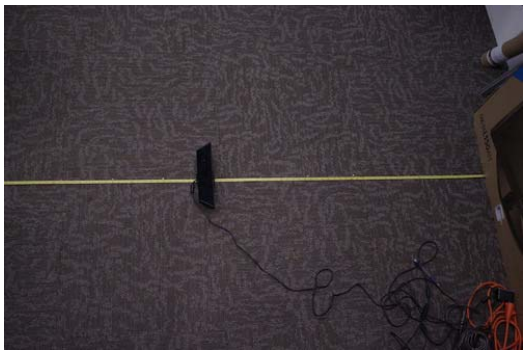
Chapter 5

Experimental results

In this chapter we present the reconstructed results from our novel point clouds registration approach. We evaluate the reconstructed results on qualities and quantities. Before we perform scanning the indoor environments, we first build a table comparing real depths with raw depths. We then test different truncated distances constructing TSDF structure. We also use the estimative virtual camera location with respect to the real camera location for comparison. In the last two section, we present our reconstructed results and show that our proposed approach can achieve better accuracy than existing SLAMs in reconstructing indoor environments with large planar surfaces.

5.1 Raw depths vs. real depths

In this section, we measure the real depths and obtain the corresponding raw depths from the depth camera. In Figure 5.1, we take the big brown box as the original point. The depth camera gradually move backward and detect the corresponding depth values. The software we capture the depth images is OpenNI. The measurement results are shown in Figure 5.1. In our experiment, the unit for depths is millimeter.

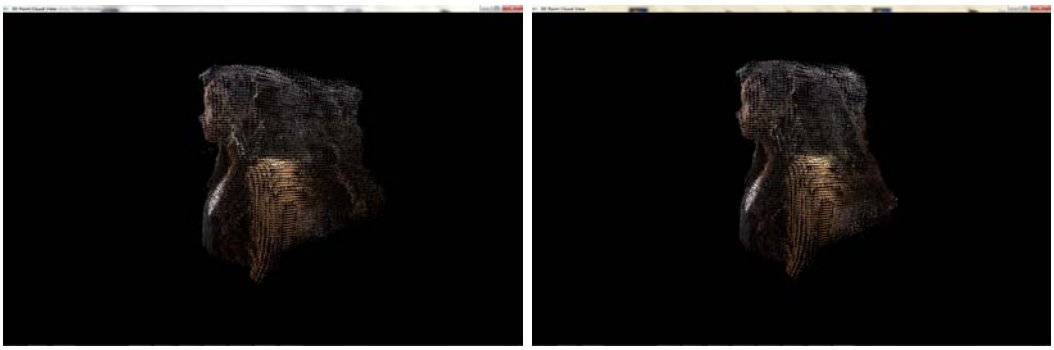
Physical depth value	Raw depth value
600	603
800	806
1000	1006
1200	1219
1400	1413
1600	1626
1800	1833
2000	2027

Figure 5.1: Real depths versus raw depths

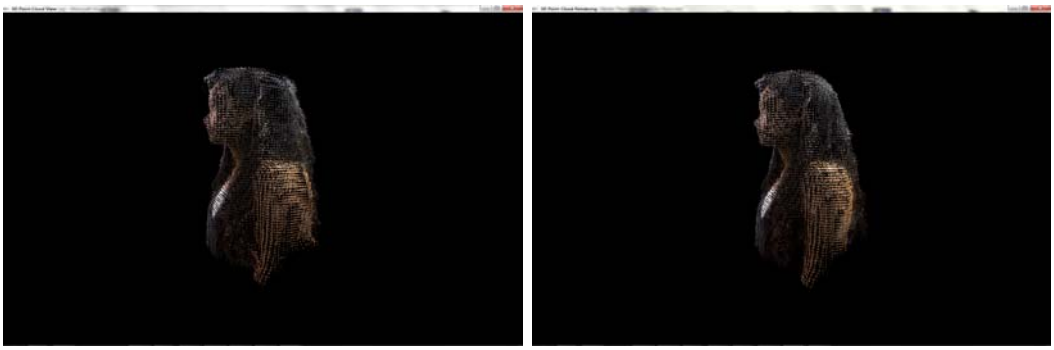
The result shows that the ratio between the real depth and raw depth is roughly 1 to 1 when the real depth is less than 2000. None of the depth value exceeds 2000 during the experiments when scanning the 3D environments.

5.2 Constructing TSDF structure on different truncated distances

In this section, we test different negative truncated distance ν on the constructed TSDF voxel structure. The voxel which is beyond ν will not be updated. The constructed TSDF structure is used to estimate the surface points when the virtual camera ray casts the TSDF structure. We perform the 360° human body scanning and observe the reconstructed results with different ν. The voxel size is $8 \times 8 \times 8$ mm, and $W_n(V) = 1$. In Figure 5.2a, we test ν = -300 and find the result having



(a) $\nu = -300$ (b) $\nu = -200$



(c) $\nu = -100$ (d) $\nu = -30$

Figure 5.2: Reconstructed human bodies from the different truncated distance

serious misalignment. We then gradually test ν from -300 to -30 and find the errors on the reconstructed human body continuously decrease. Figure 5.2d shows that the point clouds are aligned correctly with ν = -30. We conclude that when we perform 360° human body scanning starting from the front side and passing $\frac{1}{2}$ circle to the back side, the voxel which is in front of the back side still has the negative TSDF value if ν is very long. It results in generating the wrong surface points. To avoid this problem, we need to make sure negative truncated distance is inside the human body from perspective view. Therefore, we only store the voxels' TSDF values which are sufficiently closed to the surface.

5.3 Virtual Camera localization measurement

The accuracies of the virtual camera's locations influence the 3D reconstruction results. We perform the 360° scanning on the dummy and record the tracking locations. To make sure the RGB-D camera is stable while moving, we mounted the camera on the tripod wheeled trolley. The torpedo level is also put on the tripod to ensure the depth camera is parallel to the floor. The camera doesn't move in Y dimension during the entire scanning. We record tracking locations by selecting 8 spots with the color labels. From 0° to 360°, each 45° sets a recording point. The laser pointer is hung on the tripod and used to identify whether the camera passes through recording point. The scanning equipment are shown in Figure 5.3a and 5.3b.

(a) Tripod wheeled trolley (b) Torpedo level and laser pointer

Figure 5.3: 360° scanning equipment

Note that the direction in X dimension is reversed in all depth and color images taken by the RGB-D camera during the experiments. When the camera is moving toward

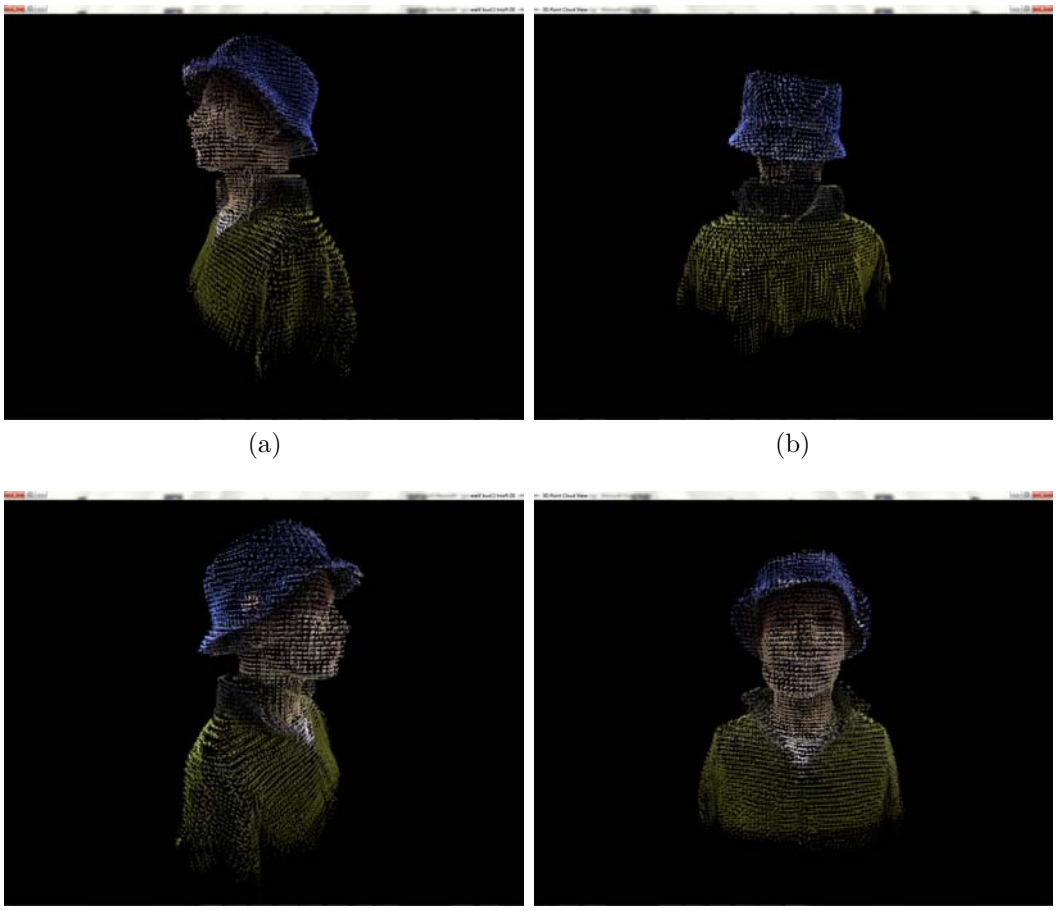
Table 5.1: Ground truth location versus virtual camera location

	Physical location (cm)	VR's location (cm)	MSE
Original Point (0°)	(0, 0, 0)	(0, 0, 0)	0.0
1/8 circle (45°)	(56.5, 0, 28)	(57.3, 0.2, 27.7)	0.8775
1/4 circle (90°)	(70.5, 0, 82)	(71.6, 0.92, 81.4)	1.5545
3/8 circle (135°)	(47, 0, 151.1)	(48.1, 1.8, 149.6)	2.5884
1/2 circle (180°)	(7, 0, 164.5)	(5.6, 2.1, 166.4)	3.1591
5/8 circle (225°)	(-60, 0, 144.5)	(-61.9, 2, 148.9)	5.1933
3/4 circle (270°)	(-74, 0, 90.5)	(-78.7, 0.94, 89.2)	4.9662
7/8 circle (315°)	(-69, 0, 24)	(-72.2, 0.7, 19.1)	5.8941
1 circle (360°)	(0, 0, 0)	(0.8, -0.35, -2.4)	2.5539

positive direction in X axis, the color and depth images are actually moving toward negative direction in X axis. Table 5.1 shows that virtual camera's locations with respect to the ground truth camera locations, only having tiny errors between the virtual camera location and the ground truth camera location.

5.4 3D reconstructions on human bodies and indoor environments

In this section, we show our reconstructed result on the human bodies and 3D environments. Figure 5.4 is the result of our 360° scanning on dummy in Section 5.3, and the reconstructed dummy is viewed in different perspectives. Figure 5.5 are other human bodies scanning and viewed in different perspectives. Figure 5.6 shows the reconstructed results on the indoor environment.



(a) (b)

(c) (d)

Figure 5.4: The dummy 360° scanning: a) Camera passing $\frac{1}{4}$ circle. b) Camera passing $\frac{1}{2}$ circle. c) Camera passing $\frac{3}{4}$ circle. d) Camera passing one circle. The Camera moves in the counterclockwise direction

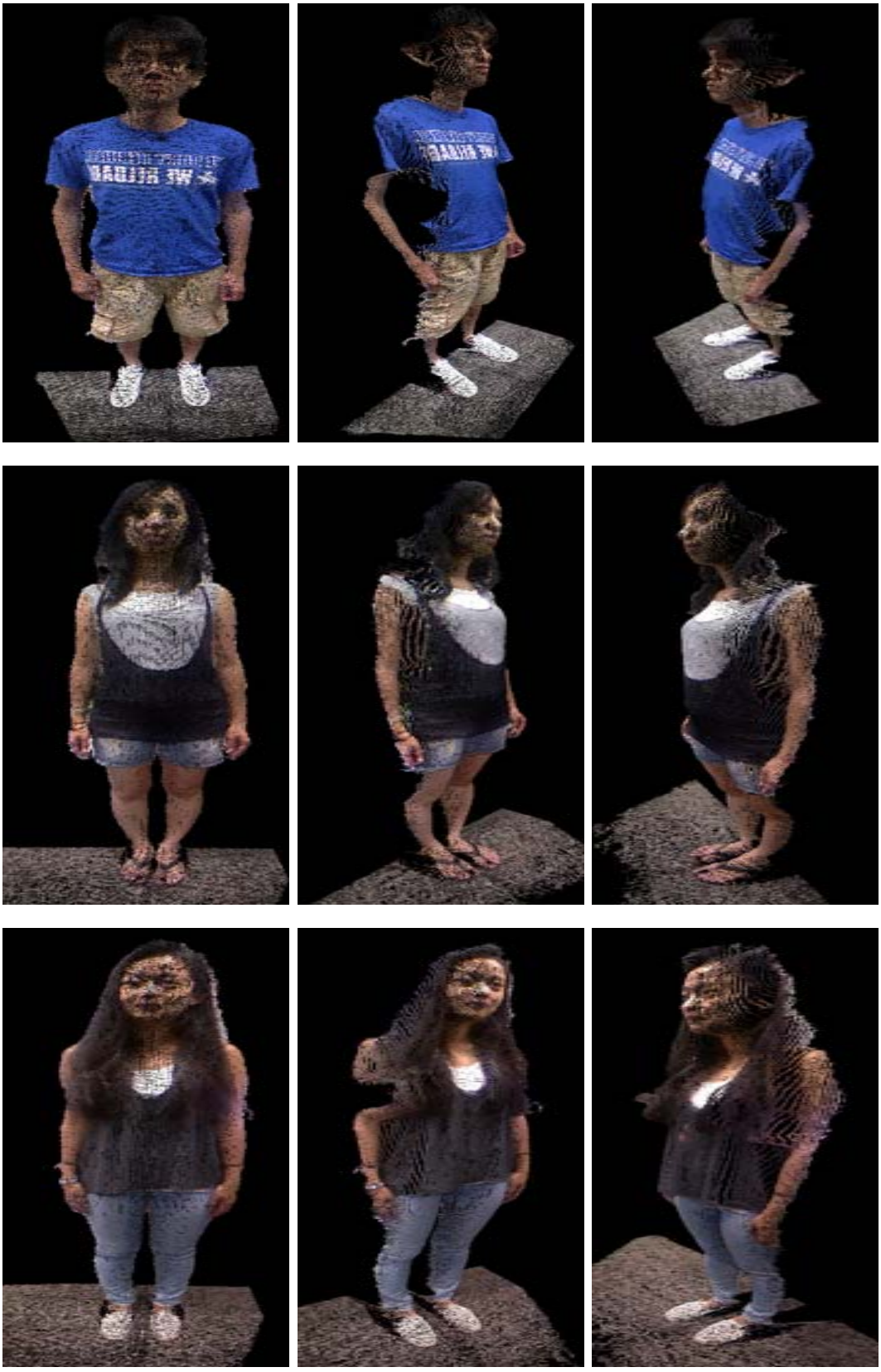


Figure 5.5: Reconstructed front side bodies

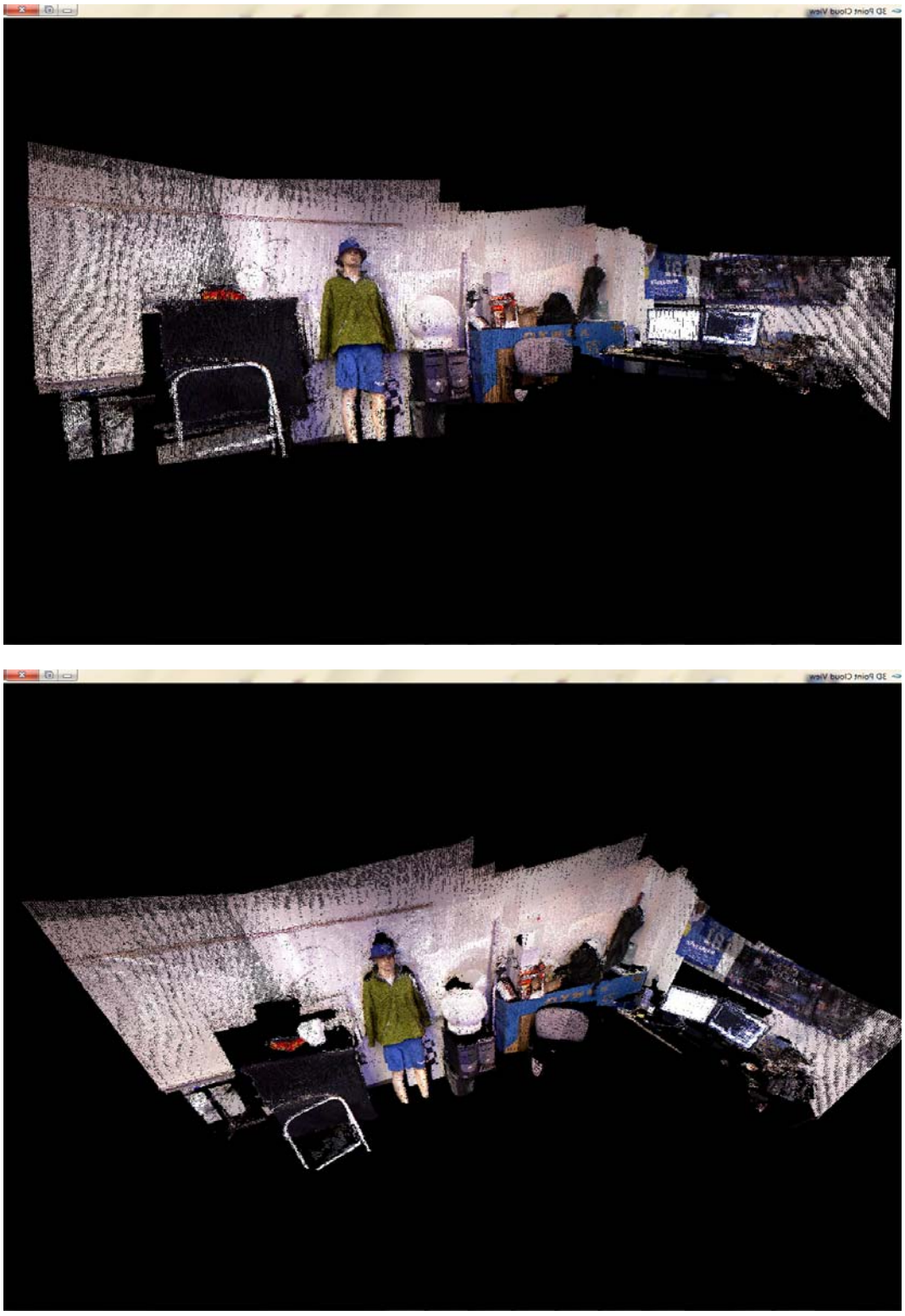


Figure 5.6: Overview of the reconstructed indoor environment

5.5 3D reconstructions with large planar surface

5.5.1 Qualitative Evaluation

Figure 5.7 is our reconstructed indoor environment with the planar surfaces, and the voxel size is $10 \times 10 \times 10$ mm. To give a better analysis of the proposed method, we concentrate on the planar areas of the scanned environment and compare the reconstructed results with the ones by [1]. The reason of choosing [1] for comparison is that it represents a relatively popular approach that has been adopted by many others [39, 40]. In Figure 5.8, the images are rendered by projecting the reconstructed 3D data to an arbitrary virtual view. Our results have significant improvements over the original scheme through a better preservation of the texture information on the planes. In particular, the text on the posters are clearly legible in the virtual views.

5.5.2 Quantitative Evaluation

For quantitative evaluation, we compare the estimated camera poses and locations against the ground truth, which is manually measured. We first use the Kinect to scan the environment against a predefined path. Along the path, we pick 10 arbitrary positions and physically measure the relative translation T and rotation R of the cameras on each spot. According to the manual measurements, a sequence of cameras are plotted in the 3D space as green cameras in Figure 5.9a. Then based on the associated frames on the spots, two sequences of transformations are estimated respectively by our proposed method and [1]. Their results are shown in the same figure. For demonstration purpose, we arbitrarily raise the blue cameras (our result)

Figure 5.7: Overview of reconstructed indoor environment with the planar surfaces

(a)

(b)

(c)

(d)

(e)

Figure 5.8: Virtual views comparisons: (a) is our improved result with respect to Figure 1.3 in introduction; (b)(d) and (c)(e) are the corresponding results by [1] and our method.

and cyan cameras ([1]) along the y axis by a fixed distance.

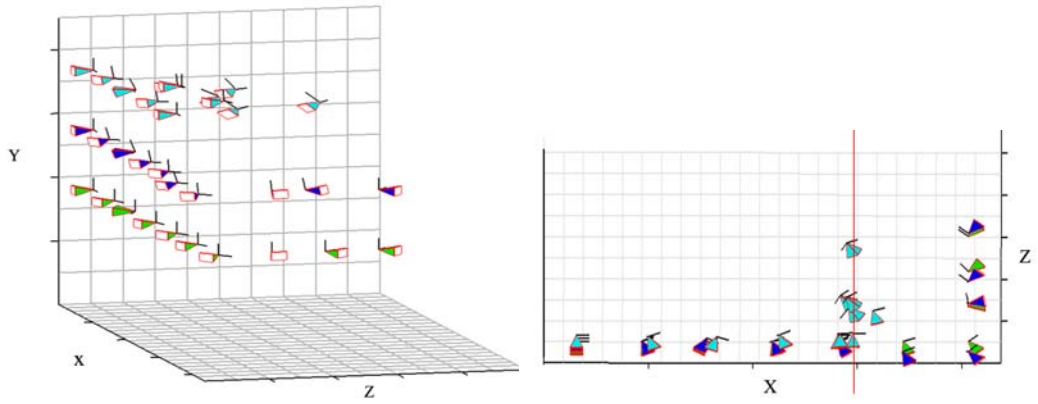
Figure 5.9b provides the top view of the results: the scan starts from the left side along the x axis and ends in the z direction. The total path is about $3.5m$. For the first few camera positions, all the three results are aligned closely due to the corresponding part of the captured environment involves considerable depth variation. After the red

Table 5.2: Estimation errors

	Category	error T	error R
By [1]	nonplanar	$0.012m$	$0.208°$
	planar	$0.731m$	$39.20°$
Ours	nonplanar	$0.009m$	$0.224°$
	planar	$0.024m$	$0.875°$

boundary, the camera enters large plane regions (the indoor wall), which causes the estimated cyan cameras start to mess up. In contrast, our results are not affected notably by the planes and remains consistent changes against the ground truth.

Table 5.2 summarizes the estimation errors: the translation error T and rotation error R are statistically computed in terms of the offsets from the ground truth when the camera is scanned with a movement of 1.0m. The analysis is conducted by two different occasions depending on whether the scanned environment has dominant plane surfaces.



(a) (b)

Figure 5.9: Camera pose estimation results: the ground truth is physically measured as the green cameras shows; the cyan cameras and blue cameras respectively indicate the results by [1] and our method.

Chapter 6

Conclusion an Future Work

In this research, we have presented a novel point cloud registration approach by using a commodity RGB-D camera reconstructing 3D indoor environments with planar surfaces. Color feature descriptors are first identified from the color images and their correspondences across different frames are robustly identified. These correspondences are then projected onto the 3D coordinate system where they undergo a second stage of noise removal. An initial camera pose transformation is finally estimated which serves as the starting point of the iterative ICP process on the depth data. All the depth data are aggregated in a voxel structure which is essential in reducing the drifting error and rendering virtual camera views. We have also demonstrated that virtual camera's locations only have tiny errors comparing with the corresponding real camera locations we measured.

In the future research, we plan to scan and reconstruct more complicated and bigger indoor environments with higher voxel resolution. In the large indoor environment sannings, we may encounter the drifting problems because we need tens of thousands of frames reconstructing the whole scene even though we only have very tiny errors aligning around a thousand frames. Since errors are accumulated when more and more frames are aligned together. The virtual camera will deviate from tracking trajectory. The globally consistent alignment approach TORO [18] could be integrated into our SLAM system and continuously refine the rigid transforma-

tion. Another direction we are interested is applying our reconstructed results on human-computer interaction interface. For example, our research group are working on virtual mirror project relieving autism children's symptom. The treatment called Video Self Modeling (VSM). The system creates virtual objects and patient themselves to redress patient's social behaviors. Finally, the memory capacity limit us reconstructing very large-scale indoor environments. The bigger space we reconstruct, the more voxels we need. In addition, while the RGB-D camera is moving and ray casting the TSDF structure, most of the voxels are useless and will take lots of time on computing TSDF. How to smartly avoid computing the useless voxels in the whole TSDF structure is challenging. We may back-project the image corners to the 3D space and form a pyramid. The voxels which are out of the pyramid range will not consider computing TSDF.

Bibliography

[1] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew 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, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[2] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[3] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, April 1992.

[4] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):820–824, August 1995.

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

[6] T. Jost and H. Hugli. A multi-resolution icp with heuristic closest point search for fast and robust 3d registration of range images. In *3-D Digital Imaging and*

Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on, pages 427 – 433, oct. 2003.

[7] H. Kjer and J. Wilm. Evaluation of surface registration algorithms for pet motion correction. May 2010.

[8] Ju Shen, Sen ching S. Cheung, and Jian Zhao. Virtual mirror by fusing multiple rgb-d cameras. December 2012.

[9] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1403–, Washington, DC, USA, 2003. IEEE Computer Society.

[10] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052 –1067, june 2007.

[11] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

[12] Javier Civera, Andrew J. Davison, Juan A. Magallón, and J. M. Montiel. Drift-free real-time sequential mosaicing. *Int. J. Comput. Vision*, 81(2):128–137, February 2009.

[13] Steven Lovegrove and Andrew J. Davison. Real-time spherical mosaicing using whole image alignment. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (3)*, volume 6313 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2010.

[14] Richard A. Newcombe and Andrew J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, pages 1498–1505. IEEE, 2010.

[15] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 2320–2327. IEEE, 2011.

[16] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter 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.

[17] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Int. Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[18] Giorgio Grisetti, Slawomir Grzonka, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS*, 2007.

[19] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *Int. J. Rob. Res.*, 30(11):1311–1327, September 2011.

[20] Hao Du, Peter Henry, Xiaofeng Ren, Marvin Cheng, Dan B. Goldman, Steven M. Seitz, and Dieter Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 75–84, New York, NY, USA, 2011. ACM.

[21] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, April 2011.

[22] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691 –1696, may 2012.

[23] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[24] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[25] A.I. Comport, M. Meilland, and P. Rives. An asymmetric real-time dense visual localisation and mapping system. In *First International Workshop on Live Dense Reconstruction from Moving Cameras In conjunction with the International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 12 2011.

[26] C. Audras, A.I. Comport, M. Meilland, and P. Rives. Real-time dense RGB-D localisation and mapping. In *Australian Conference on Robotics and Automation*, Monash University, Australia, December 7-9 2011.

[27] George Chen, John Kua, Stephen Shum, Nikhil Naikal, Matthew Carlberg, and Avideh Zakhor. Indoor localization algorithms for a human-operated backpack system.

[28] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor. Indoor localization and visualization using a human-operated backpack system. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1 –10, sept. 2010.

[29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[30] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. J. Rob. Res.*, 27(6):647–665, June 2008.

[31] John Kua, Nicholas Corso, and Avideh Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. pages 82960V–82960V–12, 2012.

[32] Brian Curless and Marc 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, pages 303–312, New York, NY, USA, 1996. ACM.

[33] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[34] John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *In Eurographics 87*, pages 3–10, 1987.

[35] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 839–, Washington, DC, USA, 1998. IEEE Computer Society.

[36] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. Feb 2004.

[37] Yan-Tao Zheng, Ming Zhao, Shi-Yong Neo, Tat-Seng Chua, and Qi Tian. Visual synset: Towards a higher-level visual representation. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, Anchorage, Alaska, U.S., 2008.

[38] K. Berthold. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[39] 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*, 18(4):643–650, April 2012.

[40] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, November 2012.

Vita

Name: Po-Chang Su

Bachelors in Electrical Engineering

Yuan Ze University, Taoyuan, Taiwan

Place of birth: Taipei, Taiwan