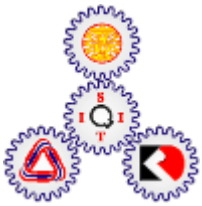


3D INDOOR RECONSTRUCTION USING DEPTH-MAP-BASED SCENE COMPLEXITY ANALYSIS GUIDED KINECTFUSION

BY

SOMKIAT KHAMPHUEA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2015

3D INDOOR RECONSTRUCTION USING DEPTH-MAP-BASED SCENE COMPLEXITY ANALYSIS GUIDED KINECTFUSION

BY

SOMKIAT KHAMPHUEA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2015

3D INDOOR RECONSTRUCTION USING
DEPTH-MAP-BASED SCENE COMPLEXITY ANALYSIS
GUIDED KINECTFUSION

A Thesis Presented

By

SOMKIAT KHAMPHUEA

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS

Approved as to style and content by

Advisor and Chairperson of Thesis Committee

(Assoc. Prof. Toshiaki Kondo, Ph.D)

Co-Advisor

(Asst. Prof. Itthisek Nilkhamhang, Ph.D)

Committee Member and
Chairperson of Examination Committee

(Pished Bunnun, Ph.D)

Committee Member

(Prof. Sato Makoto, Ph.D)

DECEMBER 2015

Acknowledgements

I would like to express my sincere gratitude to Assoc. Prof. Toshiaki Kondo, the thesis advisor, for the innumerable support. He is the kindly advisor and one of the smartest people I know. This successful of this thesis comes from his insightful guidance and patient endurance since the day we met.

I am very grateful to Asst. Prof. Itthisek Nilkamhang, the thesis co-advisor, for his excellence comment and encouragement. He has also provided helpful discussions about the research.

Beside my advisor, I acknowledge my gratitude to Dr.Pished Bunnun, thesis committee from National Electronics and Computer Technology Center (NECTEC), and Prof. Sato Makoto, thesis committee from Tokyo Institute of Technology (Tokyo-Tech) for their excellence comments and suggestions and for their thorough reading of this thesis.

I am also grateful to my family, my mother, for giving me the opportunity to follow my dreams. She is the strongest woman in my heart. I want to thank my friends ICTES batch 5^{th} for their useful information for this thesis.

Finally, this research is financially supported by Thailand Advanced Institute of Science and Technology (TAIST), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology, and Sirindhorn International Institute of Technology (SIIT), Thammasat University (TU).

Abstract

3D INDOOR RECONSTRUCTION USING
DEPTH-MAP-BASED SCENE COMPLEXITY ANALYSIS
GUIDED KINECTFUSION

by

SOMKIAT KHAMPHUEA

B.Eng., Thai-Nichi Institute of Technology, 2011

This thesis presents a novel approach for 3D reconstruction based on KinectFusion. The iterative closest point algorithm (ICP) employed in KinectFusion works well when there are sufficient 3D features in a scene to be reconstructed. Conversely, it is difficult to reconstruct simple scenes with limited 3D features such as planar structures. We propose to use visual odometry (VO), in place of ICP, when only insufficient 3D features are available in a scene. Regardless of whether there are sufficient 3D features or not, VO works well as long as the scene contains sufficient 3D features such as textures and corner points. The proposed method then automatically selects ICP or VO, depending on the complexity of the scene. The complexity of the scene is evaluated with the magnitudes of the discontinuities in surface normal vectors in depth maps. Experimental results show that the proposed method outperforms the methods based on either ICP or VO alone.

Keywords: 3D Reconstruction, KinectFusion, surface normal, scene complexity

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Nowadays, scientific and technology researches have been developed to build human-like system from their characteristic mechanisms. Perception and recognition in computer vision is difficult to perceive shapes of three-dimensional structure and recognize the objects as human's vision look-alike. Vision system in living things can effortlessly understand shape, illumination, and color, while computer vision algorithms are so uncertain. 3D Reconstruction is one kind of Computer Vision task, which is capable of capturing shape and appearance of real objects.

1.1 Simultaneous localization and mapping

Most robot visions basically rely on a single camera as human eyes to perceive 2D image information of the scene. Human has innate ability to interpret incoming image information in the human brain. Unlike, there is no kind of innate ability in robot vision application without programing it. In recent years, the researches that performs tracking the position of a robot and mapping in an unknown scene become popular. One of the famous works is parallel tracking and mapping (PTAM) (G. Klein and D. Murray, 2007), which provides a method to estimate a camera pose in an unknown scene by adapting simultaneous localization and mapping (SLAM) algorithm using RGB images taken with a hand-held camera. It performs tracking and mapping tasks in separate processes on dual-core CPU. The features are computed from consecutive frames by moving a digital camera as shown in Figure 1.1. SLAM applications are designed to suit the available resources at operational compliance which be able to utilize visual features found in the scene. PTAM allows the user holding a hand-held camera and tracking small space for AR application.

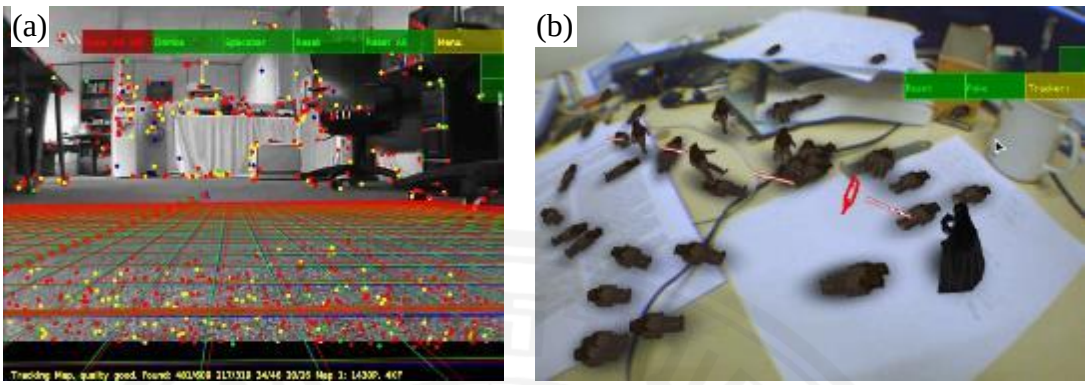


Figure 1.1 Parallel tracking and mapping (PTAM) for augmented reality application in small space (a) feature points generated after camera movement, and (b) tracking a small workspace for augmented reality application.

1.2 Dense tracking and mapping

Microsoft presents a novel camera with low-cost called Kinect camera which is based on structure light technique to obtain depth measurements. The Kinect camera consist of three major devices, IR projector, RGB camera, and IR camera. The IR projector beams the pattern of light to the environment and the pattern will be interpreted as a depth map (Smisek J et al. 2011). The live dense tracking and mapping method (DTAM) is presented by (Newcombe et al. 2011) that shows the result of tracking and mapping in real-time by using depth information instead of using 2D feature. DTAM shows the tracking performance which is more stable than PTAM. Figure 1.2 shows the live tracking comparison between DTAM and PTAM. The rapid translation causes PTAM lost in tracking while DTAM performs without re-localization.

(a) (b)

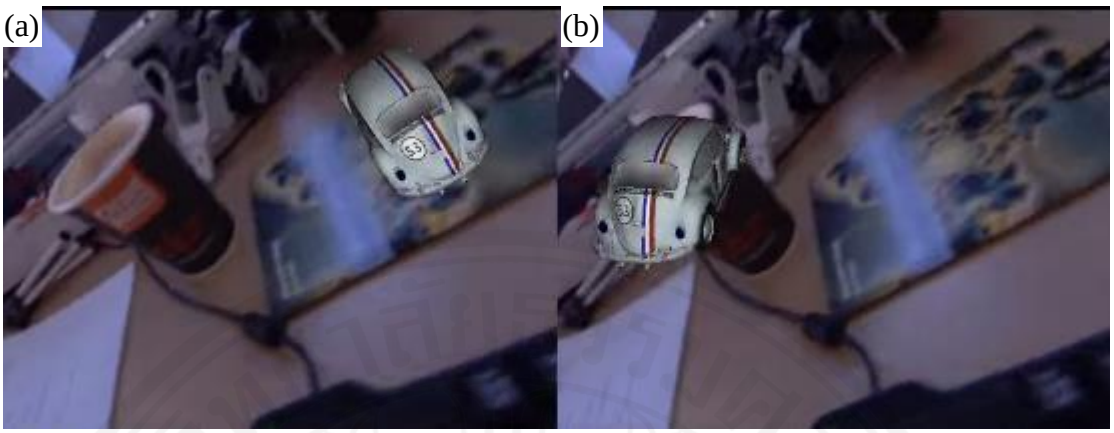


Figure 1.2 Rapid translation for live tracking comparison between DTAM (a) and PTAM (b) on small work space for augmented reality.
[Source: https://www.youtube.com/watch?v=Df9WhgibCQA]

(Newcombe et al., 2011) presents the new system called KinectFusion for building 3D scenes, that achieves detailed 3D scene reconstruction by using the truncated signed distance function (TSDF) and the iterative closest point (ICP). Meanwhile, techniques that use depth information seem to be more robust than using only RGB information but still suffer from limitations when tracking in environments with insufficient 3D features. Nevertheless, it is clear that by using both RGB image and depth information, it is possible to get a more complete, accurate and robust result. (C.Kerl et al., 2013) show robustness for camera pose estimation using the registration between two consecutive RGB-D frames where non-linear minimization is performed. (T. Whelan, 2013) also presents the combination of original KinectFusion among various visual odometry methods to provide robust tracking against planar surfaces. Visual odometry gives a better result in terms of camera trajectory estimation compared with ICP, but it is a frame-by-frame algorithm that is opposed to the drift-free concept in KinectFusion that uses ICP to build smooth 3D reconstruction results.

1.3 Motivation and objective

Over recent years, research on a dense tracking and mapping becomes popular (Wan Yi et al. 2012) with its fine reconstruction. KinectFusion cannot be achieved by a CPU, but on GPU. KinectFusion provides the algorithm resulting in detailed 3D

reconstruction which can be useful information for robotic application. 3D reconstruction is an important part of many applications, such as reverse engineering and building real-world models for virtual reality. The 3D reconstructed model can be used for a quality test in an industrial application by comparing it with the original template, such as a CAD model. More realistic 3D model for visualization can also entertain users with much more fun (Hwasup Lim et al., 2012). Our approach tries to utilize the advantage point of the two tracking methods. We first presents the modified KinectFusion method (S. Chumplue et al., 2014) by replacing KinectFusion ICP with VO (C.Kerl et al., 2013). When the original KinectFusion ICP encounters an environment with insufficient 3D features, VO gives a better result in terms of camera trajectory estimation compared with the original KinectFusion ICP, but it is a frame-by-frame algorithm that is opposed to the drift-free concept in KinectFusion that uses ICP to track and build smooth 3D reconstruction results. When the scene has sufficient 3D features for ICP, it gives a better result in terms of camera trajectory estimation compared to VO. Thus, we presents the integration method (S. Khamphuea et al., 2015) that utilizes the advantage of the two tracking methods both ICP and VO by using depth-map-based switching strategy. In Section 3 Result and Discussion, we experimentally show the result of 3D reconstruction with different types of scenes and show the result from our scene complexity analysis method. Also, qualitative and quantitative evaluations are conducted by using the ground-truth provided by (A. Handa et al., 2014).

The objective of this study is to develop a novel technique for scene complexity analysis to guide KinectFusion to tracking in insufficient 3D features. The key point of this proposed method try to analyze the scene which can be divided into two groups: complex and planar. By adding alternative algorithm, KinectFusion performs tracking in planar scene by using switching strategy.

1.4 Thesis outline

The rest of this thesis is divided into 4 chapters as follows.

Chapter 2 explains background methods. The processes in KinectFusion pipeline are describe in detail.

Chapter 3 shows the proposed method that uses an integrated approach technique between ICP in the original KinectFusion and Visual Odometry. A depth-map-based scene complexity analysis that is used to select the suitable algorithm for KinectFusion are described.

Chapter 4 discusses the experimental results from the scene complexity analysis and the integrated approach. The results is conducted using provided ground-truth and shows that the proposed method is out perform ICP and VO alone.

Chapter 5 shows the conclusion of this study based on the experimental results

Chapter 2
KinectFusion Model

KinectFusion is composed of well-known algorithms presented by (Newcombe et al., 2011). This is achieve by novel graphic card that provides parallel processing on a GPU. The goal of KinectFusion is to provide a real-time dense surface mapping and tracking using a Kinect camera. The KinectFusion algorithm are described in 4 steps as shown in Figure 2.1.

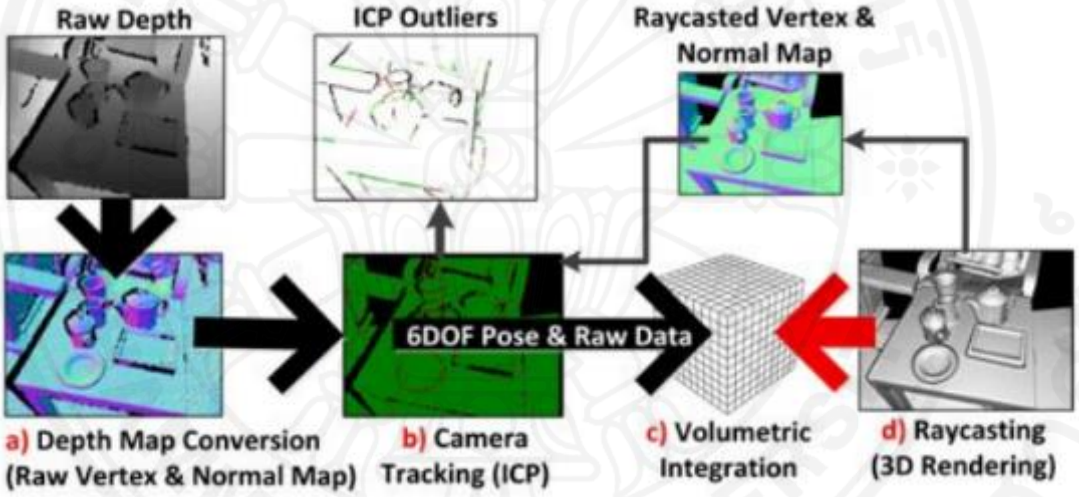


Figure 2.1 Overview of the KinectFusion pipeline presented by Newcombe, 2011.

2.1 Depth map conversion

At the beginning, the stream of raw depth maps R_k at frame time *k* are obtained from a Kinect camera. A raw depth map comprises with a noisy depth measurement $R_k(u) \in \mathbb{R}$ at each pixel u in image domain $u \in \mathcal{U} \in \mathbb{R}^2$. A bilateral filter is used to reduce the noise of a raw depth map R_k,

$$D_k = \frac{1}{W_p} \sum_{\mathrm{u}' \in \mathcal{U}} \mathcal{N}_{\sigma_1}(\|\mathrm{u} - u'\|_2)\mathcal{N}_{\sigma_2}(\|\mathrm{R}_k(u) - \mathrm{R}_k(u')\|_2)\mathrm{R}_k(u'), \tag{2.1}$$

where $\mathcal{N}_\sigma(t) = e^{-t^2\sigma^{-2}}$ and W_p is a normalizing constant. Then a filtered depth map D_k is converted to vertex map V_k by using a constant camera calibration matrix K,

$$\mathrm{V}_k(u) = \mathrm{D}_k(u)\mathrm{K}^{-1}u\,. \tag{2.2}$$

The normal map is conducted by using cross product with neighboring projected vertices,

$$\mathrm{N}_k(u(x,y)) = \text{normalize}[(\mathrm{V}_k(x+1,y) - \mathrm{V}_k(x,y)) \times (\mathrm{V}_k(x,y+1) - \mathrm{V}_k(x,y))\,]. \tag{2.3}$$

The filtered depth map will be stored at 3 pyramid levels $L = 3$. The first level $\mathrm{D}_k^{l=1}$ is set to the original filtered depth map. Each higher level D_k^{l+1} is converted to the half resolution of the previous level by block averaging. The vertex map and normal map will be calculated at each three levels by repeating equation 2.2 and 2.3 with corresponding depth map level. The result in this step is $\mathrm{V}_k^{l\in[1\ldots L]}$ and $\mathrm{N}_k^{l\in[1\ldots L]}$.

The 6 degree of freedom (6DoF) $T_{g,k}$ is represented by a rigid body transformation matrix which is a combination of a rotation 3×3 matrix $\mathcal{R}_k$ from $\mathbb{SO}(3)$ and a translation 3×1 vector t_k from $\mathbb{R}^3$. The vertex map and normal map in global coordinate can be calculated by,

$$\mathrm{V}_{g,k}(u) = T_{g,k}\mathrm{V}(u), \tag{2.4}$$

$$\mathrm{N}_{g,k}(u) = \mathcal{R}_{g,k}\mathrm{N}_k(u). \tag{2.5}$$

2.2 Camera tracking

Camera tracking is perform to track each new depth frame by estimating a new incremental rigid body transformation that closely aligns points of two consecutive

depth frames. The alignment is based on iterative closest point (ICP) algorithm. Basically, most of the tracking algorithms try to find good correspondence points and try to optimize the number of iterations by reducing the number of points. KinectFusion assumes that the motion between two consecutive frames is small by maintaining a high tracking frame-rate. This assumption allows KinectFusion to use all depth information by using fast projective data association (G. Blais and M. D. Levine, 1995). The point-plane iterative closest point algorithm (Kok-Lim Low, 2002) is employed to track a live surface vertices V_k and a predicted surface from the global model $\hat{V}_{k-1}$. This is called frame to model tracking. The goal of the point-plane iterative closest point algorithm is that it minimizes the error metric where sum of the square distance is employed. Figure 2.2 shows a principle of the error distance between two surfaces.

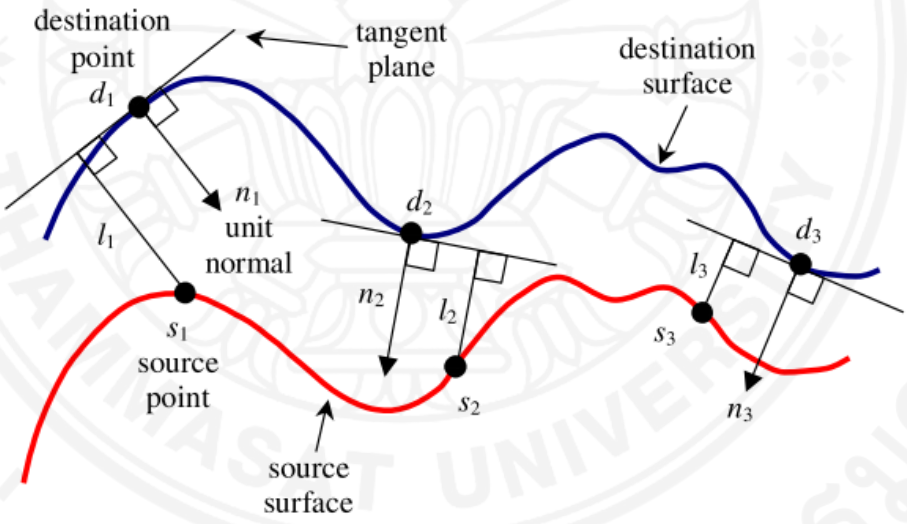


Figure 2.2 Point-to-plane error between correspondences (Kok-Lim Low, 2002).

The correspondences in KinectFusion $\{\mathrm{V}_k(u), \hat{\mathrm{V}}_{k-1}(u) | \Omega(u) \neq \text{null}\}$ can be obtained by using the fast projective data association algorithm where the distance threshold is ε_d and the different angle threshold is ε_θ.

$$\Omega(u) \neq \text{null iff} \begin{cases} \mathrm{D}_k \text{ is a valid depth value,} & \text{and} \\ \left\| T_{g,k}\mathrm{V}_k(u) - \hat{\mathrm{V}}_{k-1}(u) \right\|_2 & \leq \varepsilon_d, \text{ and} \\ \langle \mathcal{R}_{g,k}\mathrm{N}_k(u), \hat{\mathrm{N}}_{k-1}(u) \rangle & \leq \varepsilon_\theta \end{cases} \tag{2.6}$$

The camera pose estimation can be obtained by optimizing the incremental transformation $\tilde{T}$ as,

$$\tilde{T}_{opt}^{inc} = \text{argmin}_{\tilde{T}^{inc}} \sum_{\substack{u \in \mathcal{U} \\ \Omega(u) \neq \text{null}}} \left\| \left(\tilde{T}^{inc} T_{g,k-1} \text{V}_k(u) - \hat{\text{V}}_{k-1}(u) \right) \hat{\text{N}}_{k-1}(u) \right\|_2, \tag{2.7}$$

$$T_{g,k} = \tilde{T}^{inc} T_{g,k-1}. \tag{2.8}$$

Given a small motion between frames, KinectFusion follows the principle of point-plane model presented by Kok-lim low. Thus, the solution for the optimal camera pose parameter can be written in 6 parameters $(\beta, \gamma, \alpha, t_x, t_y, t_z)^{\top} \in \mathbb{R}^6$.

$$\tilde{T}^{inc} = \left[\tilde{\mathcal{R}} \middle| \tilde{t}\right] = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix}. \tag{2.9}$$

2.3 Volumetric integration

Instead of using filtered depth data, KinectFusion aims to build a detailed reconstruction where the detail could be deleted by a bilateral filter. The raw depth data will be integrated by using truncated signed distance function (TSDF). TSDF is a volumetric method that contains a registered depth measurement from frame 1 … *k*. $\text{S}_k(\text{p})$ where $\text{p} \in \mathbb{R}^3$ is the TSDF integrated up to frame *k*. The volumetric consists of a certain number of voxels in each axis stored in GPU (512^3 voxel reconstruction is used in the proposed method). The size of a volume and the number of voxels indicate the quality of the model to be reconstructed. (Curless and Levoy, 1992) introduces signed distance function (SDF) fusion. The SDF represents surface of the object where it is a zero-crossing, free space as positive values, negative values where it is behind the surface as shown in Figure 2.3.

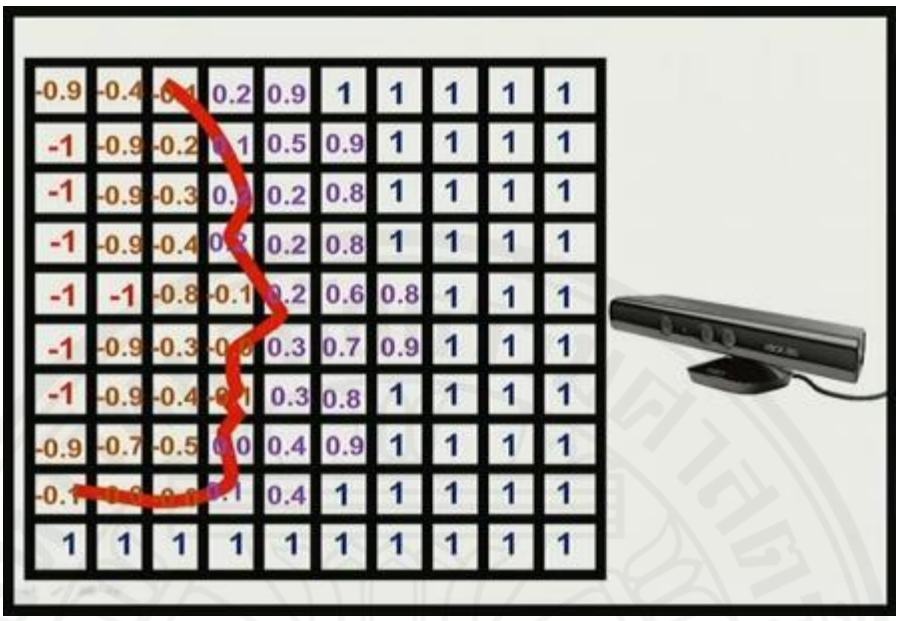


Figure 2.3 A representation of the TSDF volume grid stored in GPU. Each voxel represents the TSDF value. Zero-crossing represents the surface. The positive value represents the space where it is in front of the surface. The negative value represents wherever it is inside the surface.
[Source: http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php]

Each voxel in the TSDF stores two values: the current signed distance function $\mathrm{F}_{\boldsymbol{k}}(\mathrm{p})$ and its weight $\mathrm{W}_k(\mathrm{p})$,

$$\mathrm{S}_k(\mathrm{p}) \longmapsto [\mathrm{F}_k(\mathrm{p}), \mathrm{W}_k(\mathrm{p})]. \tag{2.10}$$

A raw depth map are fused with a known camera pose $T_{g,k}$ with a current translation $t_{g,k}$ by using projective truncated signed distance function at a point p in the global frame g. TSDF values can be calculated as Eq. (2.11). Each location of voxel can be found by using a nearest neighbor lookup ($\lfloor \cdot \rfloor$ symbol) matching the floating point to integral number. The perspective projective is calculated from the function $u = \pi(\mathrm{p})$ where $u \in \mathcal{U} = (\frac{x}{z}, \frac{y}{z})^{\top}$ in image domain and $\mathrm{p} \in \mathbb{R}^3 = (x, y, z)^{\top}$.

$$\mathrm{F}_{cur}(\mathrm{p}) = \Psi\left(\lambda^{-1}\left\|t_{g,k} - \mathrm{p}\right\|_2 - \mathrm{R}_k(u)\right), \tag{2.11}$$

$$\lambda = \left\|\mathrm{K}^{-1}u\right\|_2, \tag{2.12}$$

$$u = \left\lfloor \pi\left(\mathrm{K}\mathrm{T}_{g,k}^{-1}\mathrm{p}\right)\right\rfloor, \tag{2.13}$$

KinectFusion claims that Ψ function ensures a surface measurement (zero-crossing in the SDF) is represented by at least one non truncated voxel value in the volume either side of the surface. λ scales the measurement along the pixel ray. KinectFusion also assumes that truncated of uncertainty of a depth measurement can be done such that the true value lies within $\pm\mu$ of the measured value. Then no surface information is obtained such that a distance r from the camera where $r > (\lambda \mathrm{R_k}(u) + \mu)$ along the camera ray.

$$\Psi(\eta) = \begin{cases} \min\left(1, \dfrac{\eta}{\mu}\right)\mathrm{sgn}(\eta) & \text{iff } \eta \geq -\mu \\ null & otherwise \end{cases} \tag{2.14}$$

Every frame the new SDF and its weight are accumulated using,

$$\mathrm{F}_k(\mathrm{p}) = \frac{\mathrm{W}_{k-1}(\mathrm{p})\mathrm{F}_{k-1}(\mathrm{p}) + \mathrm{W}_{\mathrm{curr}}(\mathrm{p})\mathrm{F}_{\mathrm{curr}}(\mathrm{p})}{\mathrm{W}_{k-1}(\mathrm{p})\mathrm{W}_{\mathrm{curr}}(\mathrm{p})}, \tag{2.15}$$

$$\mathrm{W}_k(\mathrm{p}) = \mathrm{W}_{k-1}(\mathrm{p}) + \mathrm{W}_{\mathrm{curr}}(\mathrm{p}) \tag{2.16}$$

In practice, KinectFusion simply assumes that $\mathrm{W}_{\mathrm{curr}}(\mathrm{p}) = 1$ for a simple average. Weighting $\mathrm{W}_k(\mathrm{p})$ of the TSDF proportional to the uncertainty surface measurement is,

$$\mathrm{W}_k(\mathrm{p}) \leftarrow \min(\mathrm{W}_{k-1}(\mathrm{p}) + \mathrm{W}_{\mathrm{curr}}(\mathrm{p}), \mathrm{W}_\eta). \tag{2.17}$$

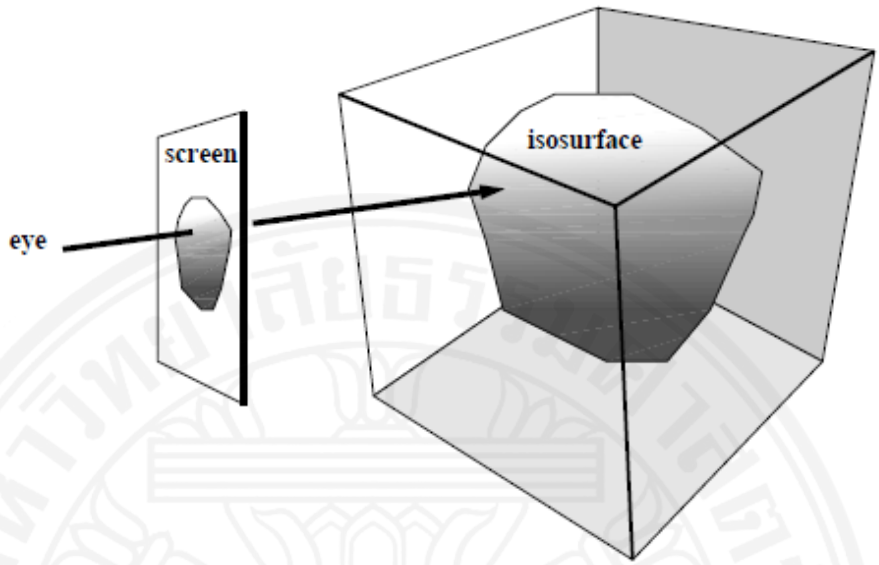


Figure 2.4 A ray casting example using marching from a camera sensor (eye) to global space (Parker et al, 1998).

2.4 Ray casting

Before the beginning of the next pose estimation, KinectFusion needs a predicted surface measurement $(\widehat{\mathrm{V}}_{k-1}, \widehat{\mathrm{N}}_{k-1})$ from a global space represented in the volumetric method. A predicted surface measurement can be obtained by using a raycast technique which a ray traverses from a camera sensor to the isosurface inside the TSDF volumetric as shown in Figure 2.4. KinectFusion simplifies a per pixel raycast algorithm presented by (S. Parker et al, 1998) by using a skipping technique based on the knowledge of the fact that near $\mathrm{F}(p) = 0$ the fused volume stores a good approximation to the SDF from p to the nearest surface interface. The steps for traversing are reduced by marching along the ray with size $< \mu$. A ray, $t = t_{g,k}\mathrm{K}^{-1}u$, is generated for each desired image pixel u and marched starting from the minimum depth for the pixel and stopping when a zero-crossing is found, or when exiting the working volume. $\mathrm{F}^{+}_{\mathrm{t}+\Delta\mathrm{t}}$ and $\mathrm{F}^{+}_{\mathrm{t}}$ are the trilinearly interpolated location which the ray intersecting to the SDF, $\mathrm{F}_k(p)$, at point p along ray t and $t + \Delta t$. A more accurate ray t^* is,

$$t^* = t - \frac{\Delta t F_t^+}{F_{t+\Delta t}^+ - F_t^+}. \tag{2.18}$$

The predicted normal maps are computed as,

$$\widehat{N}_k = \text{normalize}\big(\nabla F_k(p)\big), \tag{2.19}$$

$$\nabla F_k(p) = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right]^\top. \tag{2.20}$$

Chapter 3
Methodology

In this chapter, we describe our proposed method to solve the problem of inability of ICP tracking on insufficient 3D features in KinectFusion algorithm. ICP lost on tracking due to depth information on the scene cannot be used as the reference to align between consecutive depth maps. Additional tracking method that uses RGB image is added to prevent KinectFusion lost on tracking when the scene consists of insufficient 3D features scene.

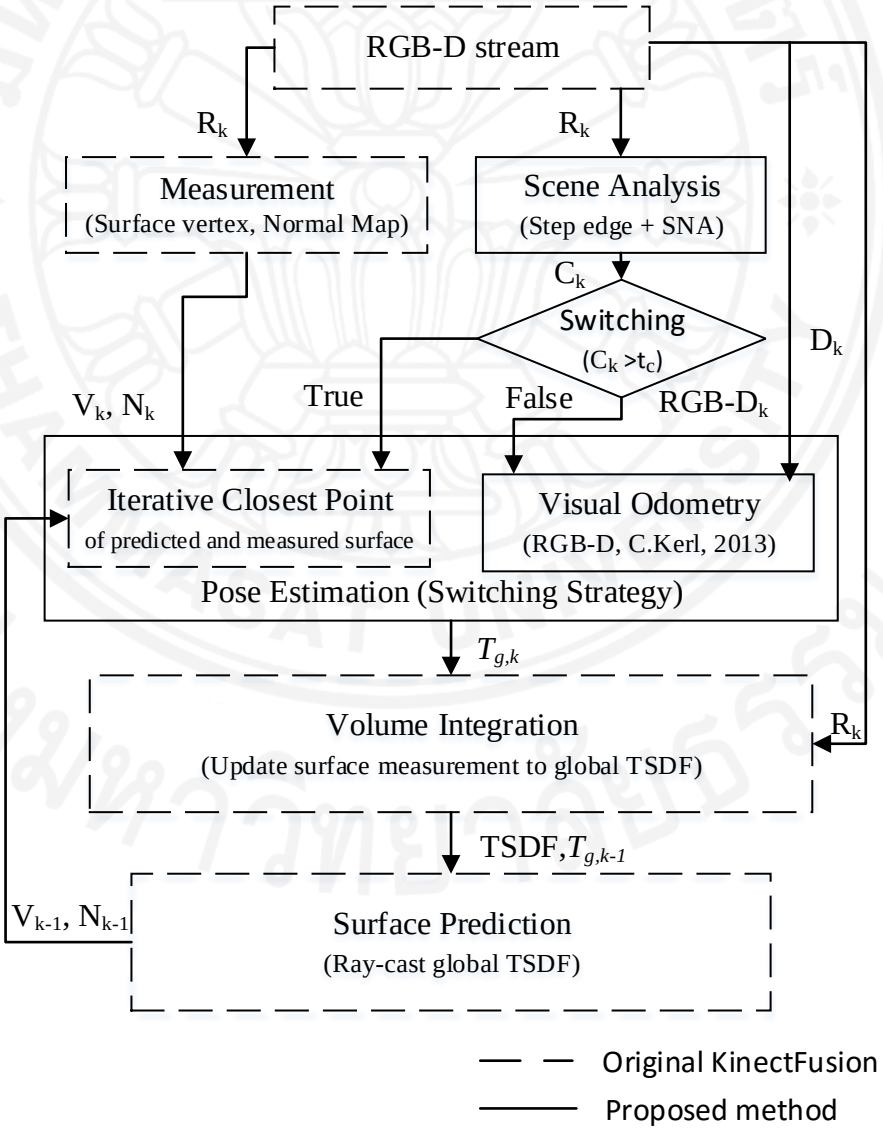


Figure 3.1 The diagram of the proposed system based on the original KinectFusion.

The proposed method combines the advantages of ICP and VO based on the original KinectFusion (Newcombe et al., 2011), which is divided into two parts: a depth-map-based scene analysis and a switching strategy. We aim to link the result of incremental 3D rigid-body transformation matrix T_k between two camera pose estimation methods using a simple scene analysis to select the most appropriate method depending on the complexity of the scene. The diagram of the proposed system is shown in Figure 3.1.

3.1 Scene Complexity Analysis

An environment that has many objects is considered as sufficient 3D features scene. The amount of complexity in a scene can be simply observed the number of objects in the scene by using method related to object detection or recognition. Normally, these techniques extremely difficult to recognize objects without any template. The proposed method presents a simplified method to compute the complexity of a scene by observing the pixels of discontinuities in a depth map responding to the boundary of objects in the scene. The KinectFusion algorithm has a registration technique called ICP which is based on a depth map registration. Assume the incoming depth map is a planar scene as shown in figure. It is difficult for ICP to register between insufficient 3D feature scenes. The proposed method aims to prevent the KinectFusion algorithm from losing in tracking when it tracks on planar structure by switching to VO algorithm as described in Section 2. The proposed method utilizes the RGB image which can be obtained from the Kinect camera. In order to prevent KinectFusion from losing tracking in planar scene. The planar scene has to be detected. The proposed method presents simply technique to calculate complexity of a depth map. These complexities describes 3D features of the scene. Usually, image features in RGB image can be obtained by SIFT, SURF, and edges detection techniques. However, these techniques offer only feature from texture which are difficult to find features when the scene has no texture.

3.1.1 Surface normal analysis

First of all, noise in a depth stream from the RGB-D camera is removed by applying a bilateral filter (C. Tomasi and R. Manduchi, 1998) as in pre-processing in

original KinectFusion paper. Secondly, we perform crease edge detection by using surface normal analysis (T. Kondo et al, 2000). We use the Sobel operators to compute the first derivatives G_x and G_y as in Eq. (3.1). The normal vector of each pixel is perpendicular to its tangent plane. We can calculate the normal of each pixel by using the cross product as in Eq. (3.2)

$$G_x(x,y) = D(x,y) * \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$
$$G_y(x,y) = D(x,y) * \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (3.1)$$

where G_x and G_y are the partial derivatives of a bilateral filtered depth image $D(x,y)$ in horizontal and vertical directions, respectively. The magnitude of G_x and G_y is calculated as in Eq. (3.2)

$$M_{gradient} = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}. \quad (3.2)$$

Secondly, we perform crease edge detection by using surface normal analysis (SNA) (Kondo, 2000) on the first derivative images G_x and G_y from Eq. (3.1). The normal vector of each pixel is perpendicular of its tangent plane, and can be calculated by using the cross product as in Eq. (3.3)

$$\begin{bmatrix} 1 \\ 0 \\ G_x(x,y) \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ G_y(x,y) \end{bmatrix} = \begin{bmatrix} -G_x(x,y) \\ -G_y(x,y) \\ 1 \end{bmatrix}. \quad (3.3)$$

The unit surface normal vector $n(x,y)$ is calculated as in Eq. (3.3)

$$n(x,y) = \begin{bmatrix} n_x(x,y) \\ n_y(x,y) \\ n_z(x,y) \end{bmatrix} = \frac{1}{\sqrt{G_x(x,y)^2 + G_y(x,y)^2 + 1}} \begin{bmatrix} -G_x(x,y) \\ -G_y(x,y) \\ 1 \end{bmatrix}. \quad (3.4)$$

3.1.2 Removing small gaps

It is difficult to distinguish the depth information of small objects from a noisy depth map. Thus, we remove the discontinuities due to all small gaps by applying a median filter to n_x and n_y components before we find the discontinuities of the surface normals in the next process. Figure 3.2 shows successfully removing small gaps information by using median filter. Figure 3.2(a) shows the RGB image of the planar scene that contains the flat wooden board. Figure 3.2(b) shows the results of scene analysis function without median filter. There are small gaps where the pink is discontinuities between the wooden board and the floor. Figure 3.2(c) shows the results of scene complexity analysis with median filter with 50×50 kernel size.

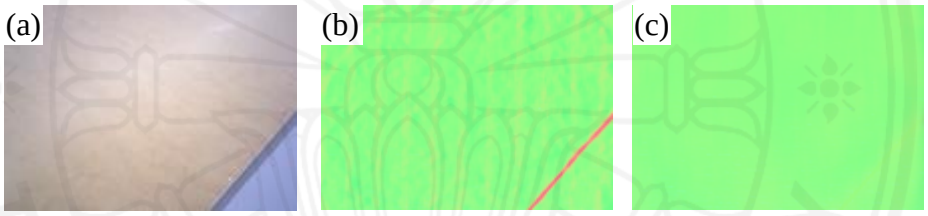


Figure 3.2 Surface normal for crease edge detection, (a) RGB image of a flat scene, (b) surface normal without median filter, and (c) surface normal with median filter.

3.1.3 Discontinuities detection

The kernel size of Sobel operators is extended from 3×3 to 9×9 as shown in Eq. (3.5). The 9×9 Sobel operators are again used to compute the first derivatives of n_x and n_y components in horizontal and vertical directions as in Eq. (3.6). We find crease edges I_n by observing discontinuities in $n(x, y)$ using its first derivatives. Note that crease edges include both roof and valley edges in a depth map.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{bmatrix}, \begin{bmatrix} +1 & 0 & 0 & 0 & +2 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{3.5}$$

$$\begin{cases} d_{xx}(x,y) = \partial n_x(x,y)/\partial x \\ d_{xy}(x,y) = \partial n_x(x,y)/\partial y \\ d_{yx}(x,y) = \partial n_y(x,y)/\partial x \\ d_{yy}(x,y) = \partial n_y(x,y)/\partial y \end{cases} \tag{3.6}$$

The magnitude of a crease edge can be computed as in Eq. (3.7). When the magnitude of the crease edge is larger than the predetermined threshold value t_n, we consider that the pixel of interest has a crease edge as shown in Eq. (3.8).

$$S(x,y) = \sqrt{d_{xx}^2 + d_{xy}^2 + d_{yx}^2 + d_{yy}^2} \tag{3.7}$$

$$I_n(x,y) = \begin{cases} 1, & S(x,y) > t_n \\ 0, & otherwise \end{cases} \tag{3.8}$$

$$C_k = \sum_{x,y} I_n(x,y) \tag{3.9}$$

Finally, the cost of scene analysis function C_k at the frame time k can be calculated from the total number of pixels corresponding to the crease edges as in Eq. (3.9). Judging from the value of C_k, we will distinguish sufficient 3D features scenes or not.

3.2 Camera pose estimation with switching strategy

(Newcombe et al., 2011) presents KinectFusion, which is a system for accurate real-time mapping of complex and arbitrary indoor scene in variable lighting condition. It gives high-detail 3D reconstructed model by allowing users to hold and move the camera freely in 6DOF. We use the KinectFusion algorithm as the key component to reconstruct 3D model in real-time by employing a combination of various well-known techniques, which are iterative closest point (ICP) and truncated signed distance function (TSDF). ICP method is used to minimize the sum of the squared distance between each source point and the tangent plane at its corresponding destination using depth data which were computed as the depth map and normal vector. By maintaining

a high tracking frame-rate, small motion from frame to frame is assumed. KinectFusion uses fast projective to obtain correspondence and the point-plain metric for pose optimization. The result of pose estimation in KinectFusion gives incremental transformation $\tilde{T}_k$ at current frame time k

$$\tilde{T}_{ICP,k} = [\tilde{R}_k|\tilde{t}_k] = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix}. \tag{3.10}$$

The incremental transformation $\tilde{T}_k$ will be applied to global transformation to obtain the latest global transformation as described in Eq. (2.8). However, the original KinectFusion has limitations. ICP will match the current depth map against a ray casting surface from global model. Matching between the two data which contain depth information from planar surface causes ICP to fail in tracking. The tracking algorithm of the original KinectFusion based on ICP is replaced by the visual odometry (C.Kerl et al., 2013) to estimate the camera motion by aligning correspondence points between two consecutive RGB-D images using sum-of-squared differences and minimizing the photometrical error between them. The advantage of this method is that it gives precise estimated motion on low 3D features environment. By using RGB images to estimate the camera pose, we can reduce the failure rate of KinectFusion when tracking on low-feature 3D geometry. This method offers using matching technique to find the camera motion parameter ξ by minimizing the photometric error between consecutive images. More information about concept and theory can be found in (C.Kerl et al., 2013) publication. After minimizing the photometric error, the Lie algebra parameter are given by $\xi = (v_1, v_2, v_3, \omega_1, \omega_2, \omega_3)$ where v_1, v_2, v_3 is the translational velocity and $\omega_1, \omega_2, \omega_3$ is the rotational velocity. Thus, rotation matrix $\tilde{R}_k$ and translation vector $\tilde{t}_k$ at the frame time k can be calculated from Lie group $\mathbb{SE}_3$ with the exponential map $\exp(\xi)$.

$$exp: \mathbb{SE}(3) \rightarrow SE(3);\ \xi \rightarrow T \tag{3.11}$$

The incremental transformation of visual odometry technique $\tilde{T}_{VO,k}$ is shown in Eq. (3.12)

$$\tilde{T}_{VO,k} = [\tilde{R}_k|\tilde{t}_k] = \exp(\xi). \tag{3.12}$$

The switching strategy combines two techniques, ICP from the original KinectFusion and visual odometry to compute global rigid-body transformation matrix $T_{g,k}$ at the frame time k. The value from the scene analysis function C_k is computed in real-time and it will select a suitable algorithm, as in Eq. (3.13).

$$T_{g,\boldsymbol{k}} = \begin{cases} T_{g,k-1}\,\tilde{T}_{ICP,k}, & if\ C_k > t_c \\ T_{g,k-1}\,\tilde{T}_{VO,k}, & otherwise \end{cases}. \tag{3.13}$$

The proposed method performs tracking operation with scene analysis. Two threshold values are selected experimentally, as shown in the result and discussion section. The global camera pose of switching tracking $T_{g,k}$ will be used in the volume integration of original KinectFusion algorithm as shown in Figure 3.1.

Chapter 4
Results and Discussions

4.1 Hardware and software specification

We have conducted experiments on a standard laptop PC with Windows 8.1 64 bits, Intel Core i7-4700MQ 2.4GHz CPU, 8GB DDR3 RAM and NVidia GeForce GT750m GPU card. The input data are retrieved by a single standard RGB-D Kinect camera with 640x480 resolution and frame-rated up to 30 Hz. The proposed method implementation is based on the large-scale KinFu project from the open-source called point-cloud library (PCL) (Radu Bogdan Rusu and Steve Cousins, 2011).

4.2 Datasets and benchmarks

We run experiments based on provided dataset ICL-NUIM (A. Handa et al., 2014) and TUM RGB-D dataset (J. Sturm et al., 2012). The qualitative and quantitative results are evaluated from these datasets in the offline process. The ICL-NUIM dataset provides synthetic RGB-D datasets by using persistence of vision raytracer (POV-ray). POV-ray is a rendering technique that calculates the way rays of light travel from a simulated camera trajectory in predefined scene to obtain RGB images and depth maps. Figure 4.1 shows a predefined scene as a ground truth model. Figure 4.2 shows the synthetic RGB-D data from the predefined scene. By using the synthetic data as RGB-D inputs, the reconstructed model in the proposed method can be compared with the ground truth model. The error distance between the reconstructed model and the ground truth model is used to evaluate the quality of the reconstructed model by using CloudCompare software. Furthermore, The TUM RGB-D dataset provides 8 datasets which are clearly categorized and divided into two groups: structure-datasets (complex scenes) as show in Figure 4.3 and no-structure-datasets (planar scenes) as shown in Figure 4.4. We use this dataset to calculate an accuracy of the scene complexity analysis method.

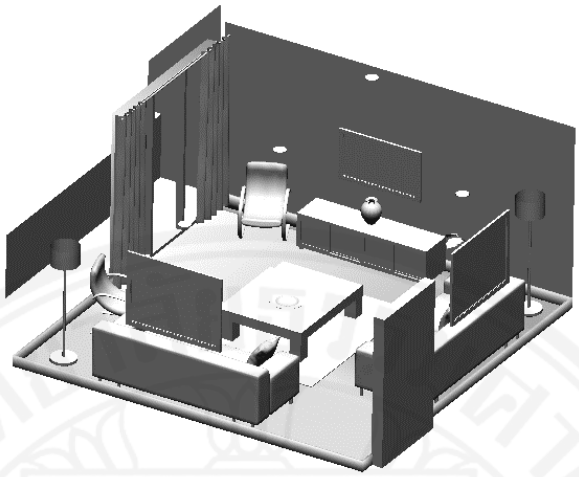


Figure 4.1 Living room as a predefined scene for POV-ray from ICL-NUIM dataset.

Figure 4.2 Synthetic screenshot of living room (Kt0 dataset) from a predefined scene as a ground truth from ICL-NUIM dataset.

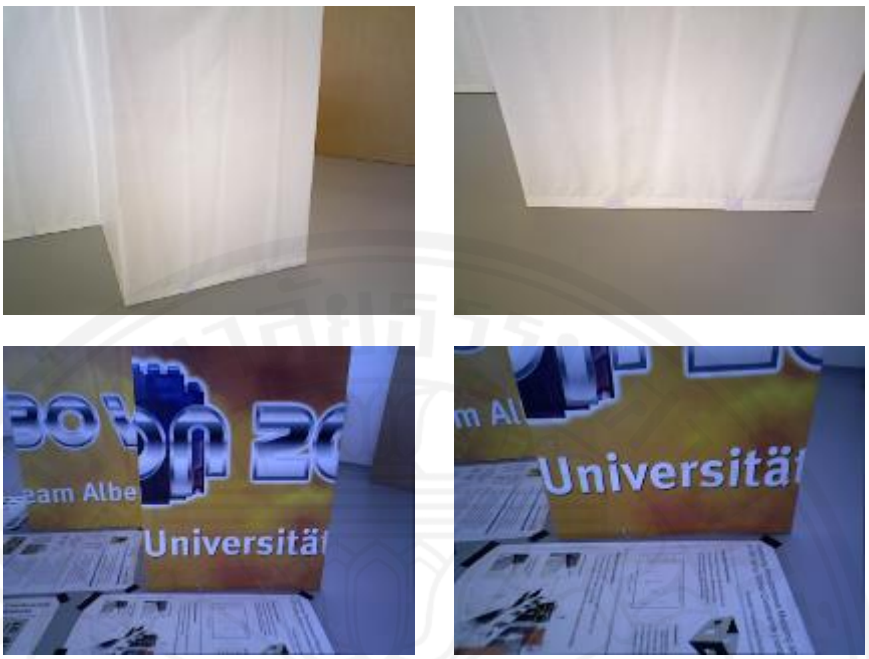


Figure 4.3 RGB images of structure-dataset from TUM RGB-D dataset.

Figure 4.4 RGB images of no-structured-dataset from TUM RGB-D dataset.

4.3 Depth-map-based scene complexity analysis

4.3.1 Surface normal analysis

Figures 4.5 and 4.6 demonstrate the process of scene complexity analysis on the TUM dataset. Figures 4.5(a) and 4.6(a) shows RGB full color images, while Figures 4.5(b) and 4.6(b) show the depth maps corresponding to the RGB images. Figures 4.5(c) and 4.6(c) then show the surface normal vectors of the depth maps. The orientations of the surface normal vectors are represented in different colors. The boundaries between different colors indicate the discontinuities of surface normal vectors, which are called crease edges. Finally, the white pixels in Figures 4.5(d) and 4.6(d) exhibit the boundaries between different surfaces. We then classify the scene to complicated or simple, depending on the number crease edges detected. The number of edges in Figure 4.5(d) is 13,118. By contrast, Figure 4.6(d) shows fewer edge points, 0, and is classified as simple.

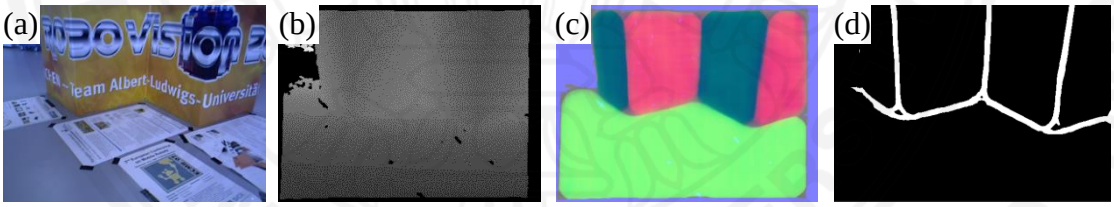


Figure 4.5 Living room dataset with *objects* presented in the scene,(a) and (b) are RGB-D images, (c) surface normal image, and (d) cost of the scene analysis where $C_k = 13{,}118$.

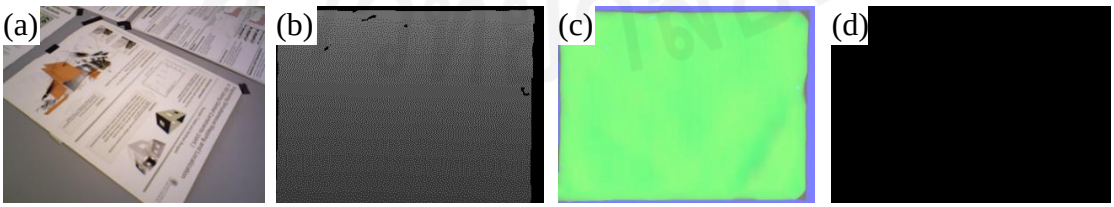


Figure 4.6 Living room dataset with *no object* presented in the scene,(a) and (b) are RGB-D images, (c) surface normal image, and (d) cost of the scene analysis where $C_k = 0$.

In practice, a depth map tends to be noisy and this affects surface normal vectors. The discontinuities in surface normal vectors are caused not only by noise but also by a rough surface of a plane. To cope with the small fluctuations in a depth map, the median filter is used to smoothen normal surface vectors with appropriated threshold value t_n as described in Eq. 3.8. Through the experiment, we have selected $t_n = 3.0$ as the threshold value. If surface normal vectors are computed without median filter, the detection of the discontinuities in surface normal vectors are too sensitive to noise. Figures 4.7(a) and (b) show the result of scene complexity analysis and the surface normal vectors without median filtering. Figures 4.7 (c) and (d) show the result of scene complexity analysis and the surface normal vectors with median filtering.

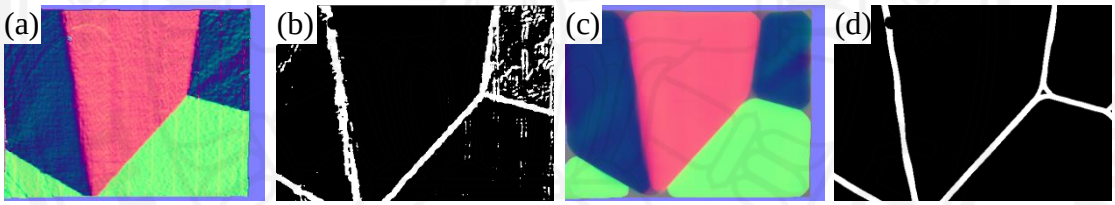


Figure 4.7 Comparison of surface normal vectors with and without using median filter, (a) surface normal without median filter, (b) scene complexity analysis with $t_n = 3.0$ and without median filter, (c) surface normal with median filter, and (d) scene complexity analysis with $t_n = 3.0$ and with median filter.

We then evaluate the scene analysis function by computing a confusion matrix which is conducted based on the ground truth from TUM RGB-D dataset (J. Sturm et al., 2012). The TP (true positive) is the scene successfully classified as a complex scene. The TN (true negative) is the scene successfully classified as a planar scene. The FP (false positive) is the planar scene wrongly classified as a complex scene. The FN (false negative) is the complex scene wrongly classified as a planar scene. Table 4.1 shows a confusion matrix by using t_c= 1000 to distinguish planar or complex scenes. Table 4.2 shows the accuracy, sensitivity, specificity, and precision of the scene analysis function from the confusion matrix following Eqs. (4.1) to (4.4).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{4.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.4}$$

The proposed method distinguishes planar scenes from complex scenes successfully. The proposed method focuses on the sensitivity performance because it prevents KinectFusion losing in tracking. There are only three complex scenes that are wrongly classified as planar scenes. The errors occurred because the view point from the RGB-D camera gives missing data or invalid pixels. The missing data causes loss of information of the crease edges, resulting in lower costs of C_k.

Table 4.1. Confusion matrix of scene analysis function using 8 datasets

		Scene analysis function result (frames)	
		Complex Scene	Planar Scene
Ground Truth (frames)	Complex Scene	3809	3
	Planar Scene	0	3638

Table 4.2. Scene analysis performance using 8 datasets

Accuracy	Sensitivity	Specificity	Precision
99.96%	99.92%	100%	100%

4.3.2 Switching strategy

The switching technique in the proposed method requires a threshold value t_c to distinguish planar scenes from complex scenes. Figure 4.8 shows a histogram of all the cost values from the scene analysis function using the 8 datasets from TUM RGB-D datasets. Figure 4.9 shows a plot of the cost values using the scene analysis function

from the 8 datasets. We experimentally select a threshold value t_c which is equal to 1000. We next investigate the step switching from VO to ICP (i.e., planar to complex scenes). By using only a single threshold value t_c= 1000, ICP still loses in tracking. It is necessary to wait until the scene has more 3D features. Mostly, the cost values of planar scenes are close to zero and the cost values of complex scene are larger than 4000. Thus, we have a wide gap of the cost values between planar and complex scenes. We experimentally define another threshold value t_c= 6000 to make ICP more stable. The proposed method uses double threshold values t_{c1}=1000 and t_{c2}= 6000. The threshold value t_{c1} is used when we switch from VO to ICP. On the other hand, t_{c2} is used when we switch from ICP to VO.

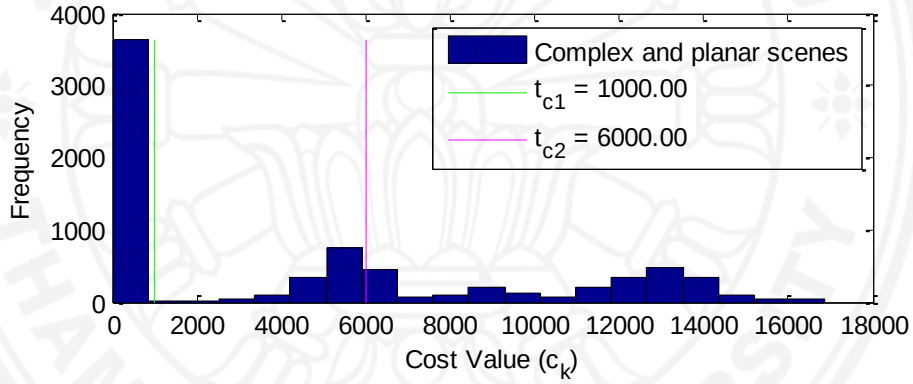


Figure 4.8 Histogram of all the cost values from the scene analysis function using 8 datasets

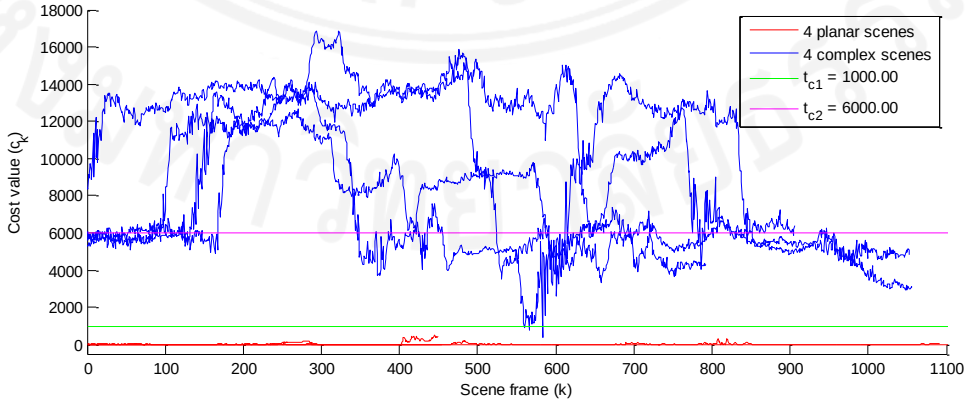


Figure 4.9 Plot of all the cost values from the scene analysis function using 8 datasets

4.3.3 Reconstruction results

We compare 3D reconstruction results using KinectFusion with different algorithms, iterative closest point (ICP) and visual odometry (VO). The input data is recorded by using the OpenNI toolkit in a real-world situation. We record both planar structures (floors) and more complicated structures (cluttered room) in an indoor environment to investigate the characteristics of the two algorithms. Figure 4.10(a) shows RGB images, while Figure 4.10(b) shows depth maps corresponding to the RGB images. When the camera faces a planar structure, the depth information is also planar. This is a difficult scene for ICP to perform 3D reconstruction. Even in this case, however, VO can utilize textural information available in RGB images for 3D reconstruction. In this way, VO can cover ICP's weakness to the scenes with planar structures. Therefore, we introduce a switching strategy to our 3D reconstruction. We switch two algorithms, depending on the complexity of the scene to be reconstructed. When the scene is complicated and full of 3D features, ICP is used. When the scene is simple and composed of planar structures, VO is selected.

Figure 4.11(a) shows the reconstruction result of VO where the flat area (floor) is well reconstructed. Figure 4.11(b) shows a limitation of ICP where part of the flat surface is missing on the reconstructed model. Figure 4.11(c) shows the proposed method using a switching strategy where the missing flat area in Figure 4.11(b) are recovered and in complex scene are finely reconstructed. When the scene contains sufficient 3D features, we select ICP because it can reconstruct 3D scenes without drifting by registering depth maps using a global model. The VO algorithm, on the other hand, performs 3D reconstruction frame by frame, which may induce drifting. Therefore, we use VO only when the scene is lack of 3D features to minimize the negative impact from the drift.

We further investigate the properties of ICP and VO. We compare reconstruction errors using a heat map generated by the CloudCompare tool. The absolute deviations between 3D reconstruction results and the ground truth mesh (C2M) are represented in different colors in Figure 4.12. The heat map in Figure 4.12(a)

appears more blue than that in Figure 4.12(b), which means that the reconstruction errors by ICP are smaller than those by VO. This explains why we use ICP, instead of VO, when the scene has sufficient 3D features. Figure 4.13 shows three absolute trajectory errors by ICP, VO, and the proposed method. The Kt0 dataset contains many scenes which are lack of 3D features. In Figure 4.13(a), we crop to 1400 frames to make the graph readable. The large red area of ICP in Figure 4.13(a) indicates that ICP loses tracking and shows jumping errors in the estimated trajectory. The proposed method reduces these errors and recovers the planar scenes. The result of the proposed method in Figure 8(a) shows a significant reduction of the red area. Meanwhile, Figure 4.13(b) shows the result of the proposed method is that the same as the result from ICP because the scene has sufficient 3D features.

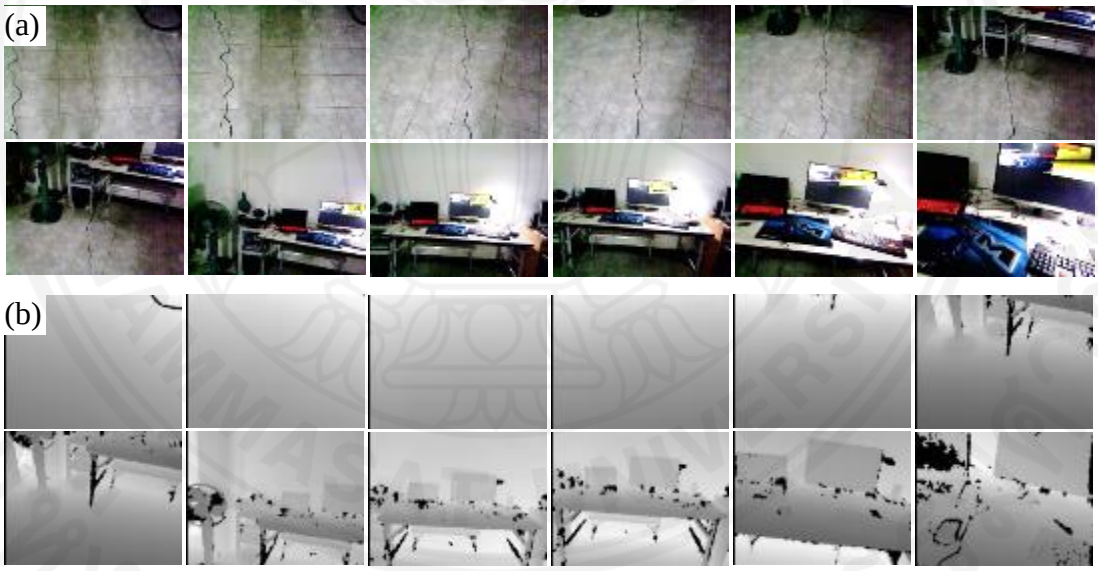


Figure 4.10 The screenshot of 198 frames from a Kinect camera in different camera poses captured by OpenNI toolkit, (a) shows RGB images which start moving along planar floor and tilt to complex scene, and (b) shows its corresponding depth images.

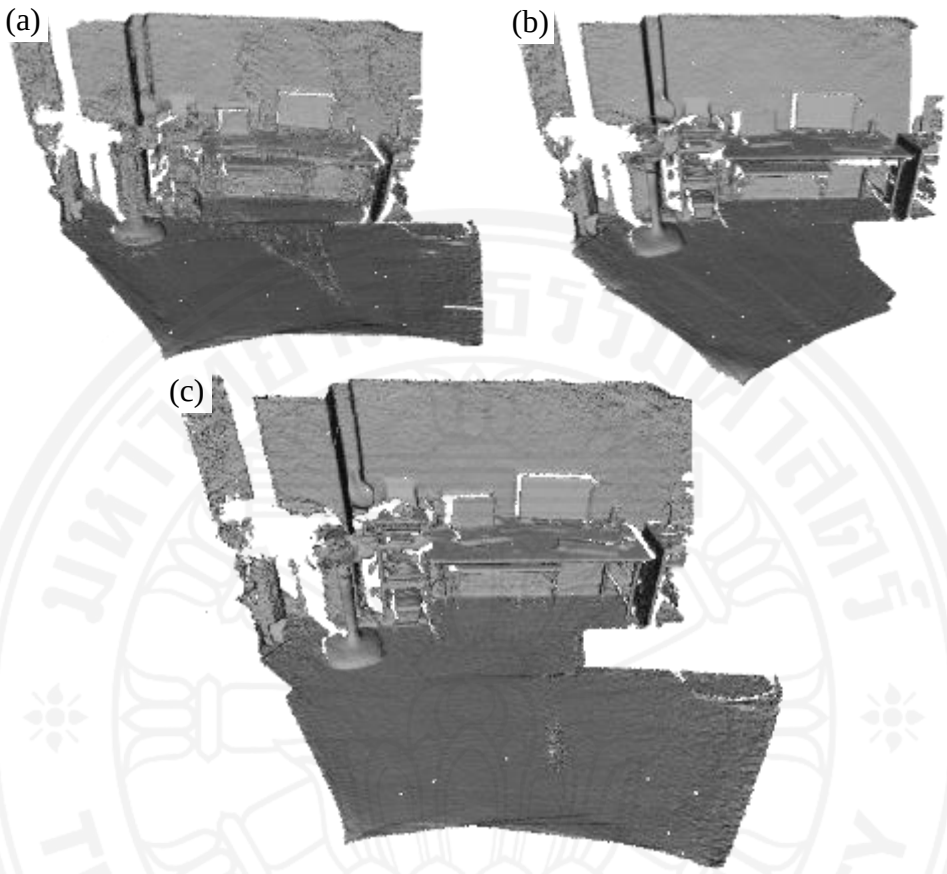


Figure 4.11 3D reconstructed results using KinectFusion with three different tracking algorithms, (a) visual odometry, (b) iterative closest point, and (c) the proposed method using switching algorithm.

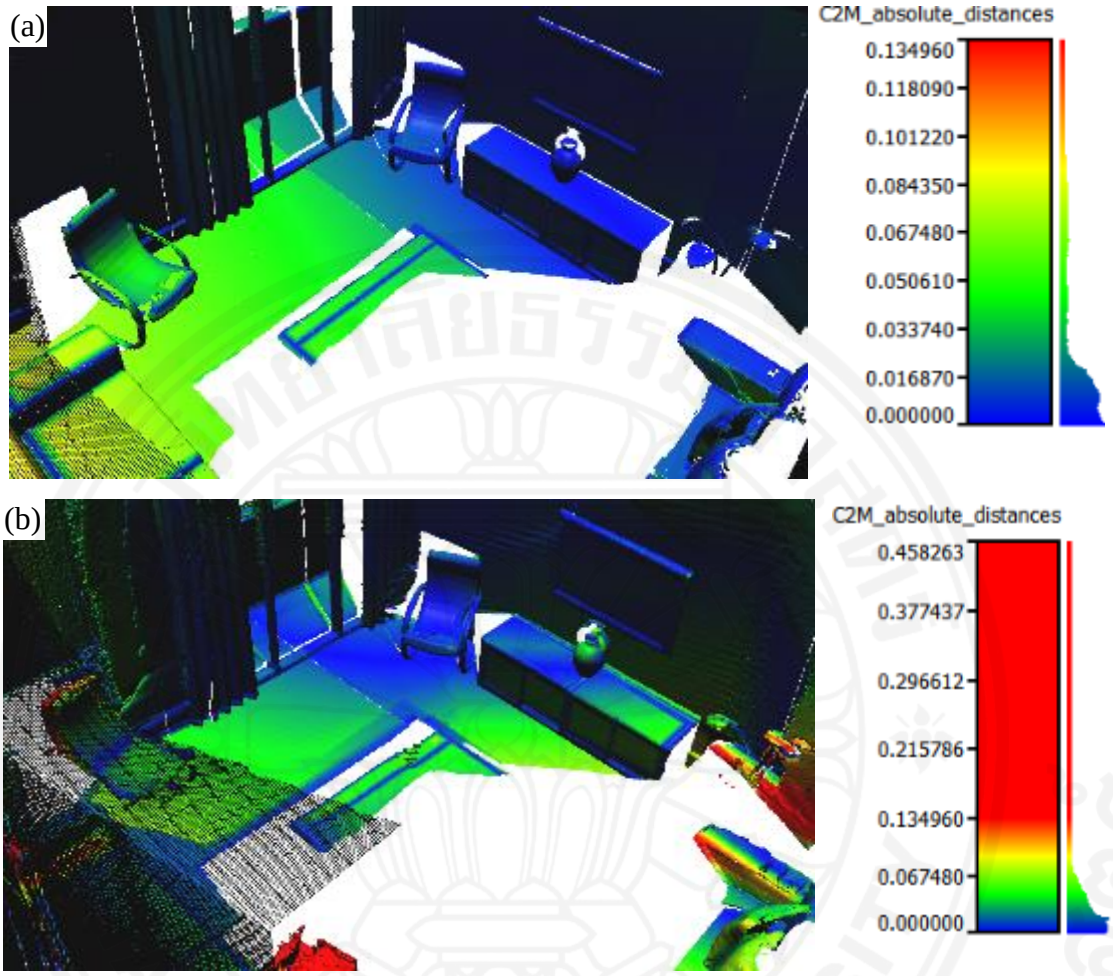


Figure 4.12 Point could to ground-truth mesh error (Heat map) of the living room Kt1 dataset, (a) using the iterative closest point algorithm, and (b) using the visual odometry algorithm.

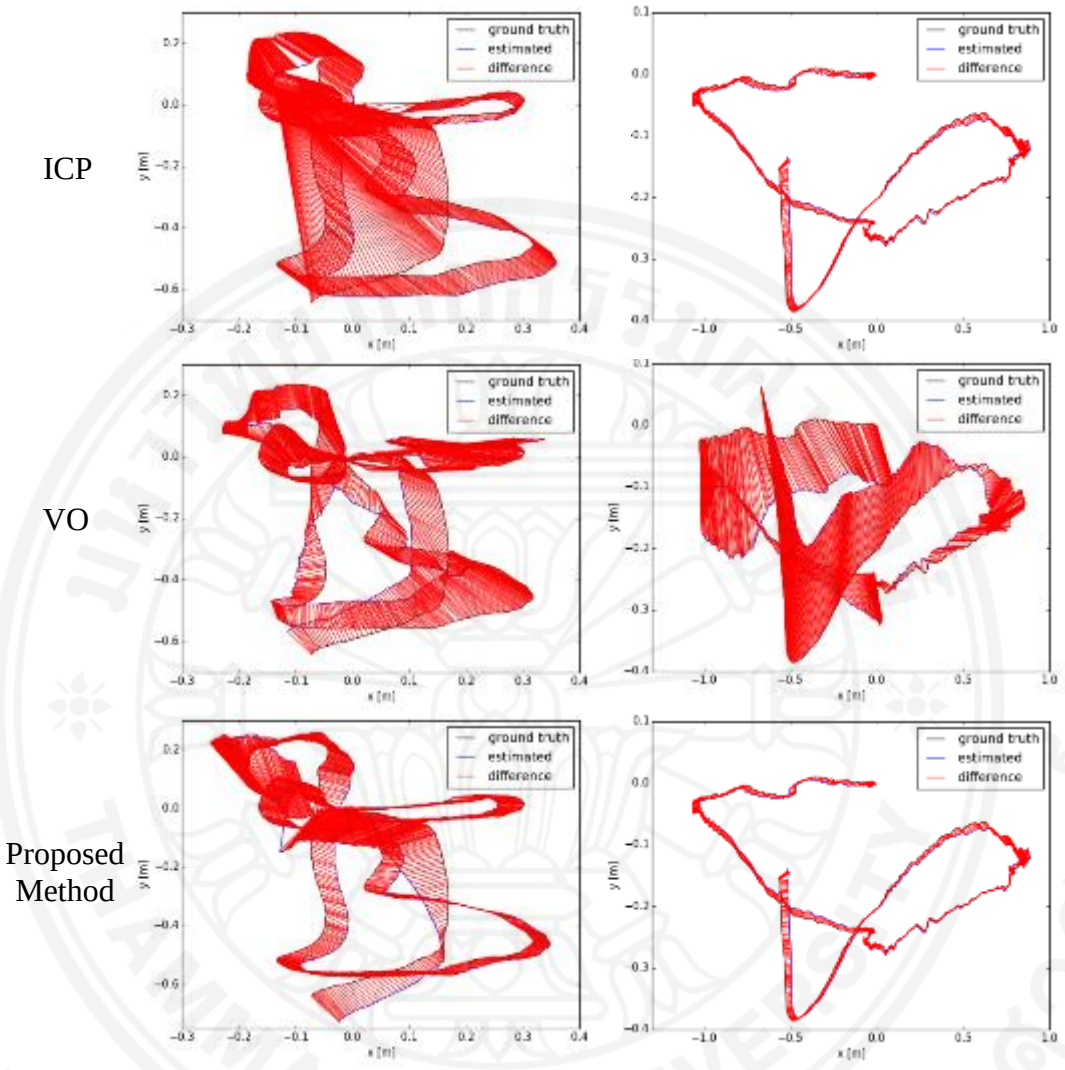


Figure 4.13 Plot of the absolute trajectory error between ground-truth and estimated camera pose for the living room (a) Kt0 (cropped to 1400 frames), and (b) Kt2 dataset.

4.3.4 Quality of reconstructed results

Unlike the VO algorithm, the original method, ICP, used in KinectFusion has an advantage of being free from drift in 3-D reconstruction. The proposed method aims to maintain this advantage, while covering the weakness of ICP, that is, its incapability of reconstructing a 3-D scene of planar structures. Table 4.3 shows the benefit of proposed method using the two methods with a switching strategy. The reconstruction errors by the proposed method on the Kt0 dataset are smaller than those by VO that performs tracking frame by frame. This verifies the earlier statement that drift-free tracking is better than frame-by-frame tracking for the scenes with sufficient 3D features. Although ICP works well for complex scenes, however, it cannot handle the scene with a planar structure. ICP tends to produce large errors that affect the final reconstructed result. The final result on Kt0 by ICP shows significant discontinuities in its 3D reconstruction. The discontinuities are so great that they cannot be used for evaluating the reconstruction errors between the reconstructed-point-cloud and the ground-truth mesh, resulting in “Failed”. Meanwhile, the results of the proposed method on the dataset Kt1 are the same as those by ICP because the dataset Kt1 comprises complex scenes only. The Kt2 dataset is not included in Table 4.3 because the scenario of Kt2 is similar to that of Kt1, and the result of the proposed method is identical to that by ICP.

We have tested the proposed method with ICP and VO algorithms on three different virtual indoor scenes using TUM RGB-D dataset format. Table 4.4 shows the differences between the reconstructed trajectories and the ground-truth trajectory. The Kt0 dataset contains scenes with a planar structure (wall), which makes ICP produce large errors of 0.69m on average. When the scene analysis step detects a scene with insufficient 3D features, the proposed method switches ICP to VO to avoid losing tracking. Meanwhile, ICP is more accurate than VO when the scene has sufficient 3D features. Therefore, in Table 4.4, the proposed method achieves the best result by integrating the two methods, ICP and VO. The key to the successful integration of the two methods lies in the switching between them. We have shown that the switch can be triggered based on the complexity analysis of the scene to be reconstructed.

Table 4.3 Statistic result of reconstructed-point-cloud compared with ground-truth mesh using CloudCompare tool.

Error (m)	Kt0 living room			Kt1 living room		
	VO	ICP	Switching	VO	ICP	Switching
mean	0.072638	Failed	0.035830	0.040537	0.033115	0.033115
median	0.053011	Failed	0.021634	0.024829	0.016384	0.016384
min	0.000000	Failed	0.000000	0.000000	0.000000	0.000000
max	0.440185	Failed	0.422587	0.458263	0.134960	0.134960
stdDev	0.073741	Failed	0.044051	0.055638	0.034402	0.034402

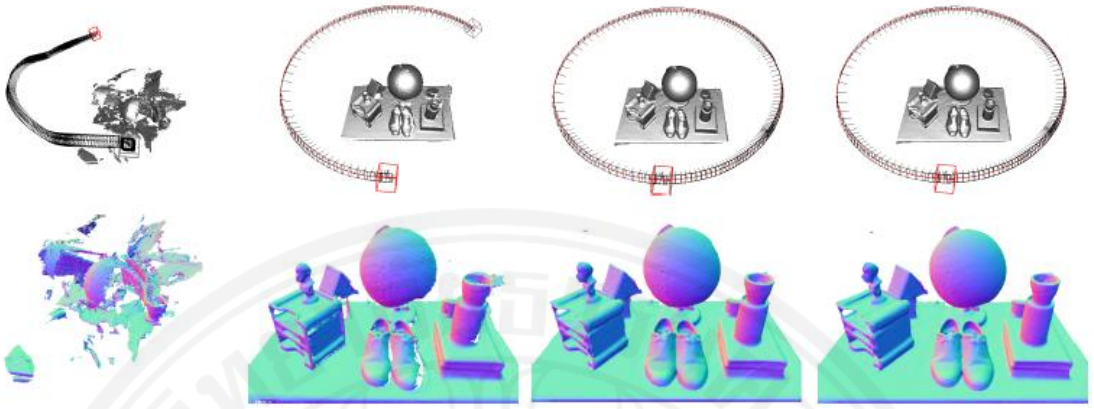
Table 4.4 Statistic result of absolute trajectory error in meters per second for virtual living room environment dataset.

Tracking error (m)		Kt0	Kt1	Kt2
ICP	RMSE	0.691711	0.046057	0.035362
	mean	0.651372	0.041396	0.032945
	median	0.531291	0.039902	0.034354
	min	0.355339	0.004393	0.005164
	max	1.222005	0.092015	0.058943
	stdDev	0.232765	0.020189	0.012849
VO	RMSE	0.212012	0.224268	0.193582
	mean	0.185164	0.179377	0.184550
	median	0.163825	0.144884	0.191285
	min	0.020245	0.050514	0.068120
	max	0.525074	0.589245	0.321130
	stdDev	0.103265	0.134612	0.058441
Switching	RMSE	0.109221	0.046057	0.035362
	mean	0.091360	0.041396	0.032945
	median	0.066319	0.039902	0.034354
	min	0.041394	0.004393	0.005164
	max	0.301264	0.092015	0.058943
	stdDev	0.059855	0.020189	0.012849

4.4 Drift-free and non-drift-free

KinectFusion describes its algorithm as a free from drifting algorithm. Figure 4.14 illustrates the advantage of drift-free tracking algorithm compared to non-drift-free tracking algorithm. Figure 4.14(a) shows the reconstructed quality of non-drift-free tracking. Figure 4.14(b) shows the reconstructed quality of drift-free tracking by partially tracking the scene. Figure 4.14(c) and (d) show the reconstructed quality of full-loop and multiple-loop drift-free tracking. By using the combination between ICP and ray casting technique, KinectFusion can produce a quality reconstructed result that free from drifting. Instead of using the RGB-D data from the previous frame, the KinectFusion algorithm uses a predicted surface measurement ($\widehat{\mathrm{V}}_{k-1}, \widehat{\mathrm{N}}_{k-1}$) in ICP method as described in Section 2.2. Thus, the error metric of ICP is computed from the predicted surface and the current surface which is the key of drift-free algorithm.

The proposed method tries to maintain the drift-free tracking by judging the scene when it has sufficient 3D features, the proposed method then switches back to ICP. Double threshold values are utilized in order to stabilize ICP as mentioned in Section 4.3.2. During the camera facing the planar structure, VO (non-drift free algorithm) is working instead of ICP. Thus, the quality of the reconstructed result depends on the accuracy of VO. When the camera estimation results in large error value, the current surface with large error will be integrated to the global model as shown in Figure 4.14(a). It is the reason that the proposed method waits the scene has more features (t_c= 6000) integrated to the global model before the proposed method generates a predicted surface and switches from VO to ICP.



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

Figure 4.14 Comparison of the reconstructed quality between drift-free and non-drift-free tracking algorithm in KinectFusion presented by Newcomb 2011, (a) non-drift-free tracking, (b) partial drift-free tracking, (c) full-loop drift-free tracking, and (d) multiple loop drift-free tracking.

Chapter 5
Conclusions

We have devised a novel approach that performs 3D indoor reconstruction by using two different camera pose estimation algorithms, the iterative closest point algorithm (ICP) and visual odometry (VO). The camera pose is estimated using ICP in the KinectFusion for scenes with sufficient 3D features. On the other hand, we switch ICP to VO when the scene is lacking in 3D features. The switch is triggered based on the complexity of the scene. We have shown that the complexity of a scene can be evaluated using the number of crease edges in a depth map. The proposed method allows the user to hold and move an RGB-D camera freely in 6DOF even when the scene is lack of 3D features that is often the case in indoor environment comprising of planar structures, such as, walls and floors. Thus, the proposed method can overcome the inability of ICP in the original KinectFusion to track a scene with insufficient 3D features. Our experimental results show that the proposed method outperforms the method based on either ICP or VO alone. For future work, the scene analysis function in the proposed method may be improved by using the hole-filling method (Lap-Fai et al. 2013) to restore missing data on the depth map.

References

Hwasup Lim, Seong-Oh Lee, Jong-Ho Lee and Min-Hyuk Sung. (2012). Putting Real-World Objects into Virtual World: Fast Automatic Creation of Animatable 3D models with a Consumer Depth Camera. **International Symposium on Ubiquitous Virtual Reality.** August 2012.

Wan Yi, Wang Jin, Hu Jingwen, Song Tiangang, Bai Yang, and Ji Zheng. (2012). A Study in 3D-Reconstruction Using Kinect Sensor. In ***Wireless Communications, Networking and Mobile Computing (WiCOM)***, *8th International Conference*, 21-23 Sept. 2012, pp.1-7.

Smisek J., Jancosek M., and Pajdla, T. (2011). 3D with Kinect. In *Computer Vision Workshops (ICCV Workshops),* ***IEEE International Conference***, 6-13 Nov. 2011, pp.1154-1160.

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

Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison. (2011). DTAM: Dense tracking and mapping in real-time. **In Computer Vision (ICCV), IEEE Int. Conf**, November 2011, pp. 2320 –2327.

Richard A. Newcombe, Shahram Izadi et. al. (2011). KinectFusion: Real-time dense surface mapping and tracking. **10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11**, Washington, DC, USA. IEEE Computer Society, 2011, pp. 127–136.

Henry Roth and Marsette Vona. September (2012). Moving volume KinectFusion. **British Machine Vision Conf. (BMVC)**, 2012.

Kok-Lim Low. (2002). Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. **Chapel Hill, University of North Carolina,** April 2002.

G. Blais and M. D. Levine. (1995). Registering multiview range data to create 3D computer objects. **IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)**, 1995.

S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. Sloan. (1998). Interactive ray tracing for isosurface rendering**. In Proceedings of Visualization**, 1998.

R. Hulik, V. Beran, M. Spanel, P. Krsek, and P. Smrz. (2012). Fast and accurate plane segmentation in depth maps for indoor scenes. In **Proceedings of Intelligent Robots and Systems (IROS)**, 2012, pp.1665-1670.

T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard and J.B. McDonald. (2012). Kintinuous: Spatially Extended KinectFusion. In **RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras.** Sydney, Australia. July 2012.

T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. (2013). Robust Real-Time Visual Odometry for Dense RGB-D Mapping In **IEEE Intl. Conf. on Robotics and Automation**. Karlsruhe, Germany. May 2013.

Christian Kerl, Jurgen Sturm, and Daniel Cremers. (2013). Robust Odometry Estimation for RGB-D Cameras, In Proc. of the **IEEE Int. Conf. on Robotics and Automation (ICRA)**, 2012.

T. Kondo, S.H. Ong, J.H. Chuah and K.W.C. Foong. (2000). Roof-edge detection in range images. **Sixth International Conference on Control, Automation, Robotics and Vision**, 2000, Paper no. 143**.**

C. Tomasi and R. Manduchi. (1998). Bilateral filtering for gray and color images. In Proc. of the **Int. Conf. on Computer Vision (ICCV)**, 1998.

A. Handa and T. Whelan and J.B. McDonald and A.J. Davison. (2014). A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. **IEEE Intl. Conf. on Robotics and Automation (ICRA)**, 2014.

S. Hinterstoisser, C. Cagniart et al. (2012). Gradient Response Maps for Real-Time Detection of Texture-Less Objects. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 2012.

Radu Bogdan Rusu and Steve Cousins. (2011). 3D is here: Point Cloud Library (PCL). **IEEE International Conference on Robotics and Automation (ICRA)**, 2011.

J. Sturm and N. Engelhard and F. Endres and W. Burgard and D. Cremers. (2012). A Benchmark for the Evaluation of RGB-D SLAM Systems. **Proc. of the International Conference on Intelligent Robot Systems (IROS)**, 2012.

Lap-Fai Yu, Sai-Kit Yeung, Yu-Wing Tai, and Stephen Lin. (2013). Shading-based Shape Refinement of RGB-D Images. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013)**, 2013, pp. 1415 – 1422.

Publications

S. Chumplue, T. Kondo, I. Nilkhamhang, P. Bunnun and M. Sato. 2014. 3D Room Reconstruction Using Visual Odometry. Guided KinectFusion with RGB-D Camera. *In* **Proceedings of the International Conference on Information and Communication Technology for Embedded Systems (ICICTES2014)**, Ayutthaya, Thailand, January 2014.

S. Khamphuea, T. Kondo and I. Nilkhamhang. 2015. An integrated approach of ICP and visual odometry for KinectFusion based on scene complexity analysis. *In* **Proceedings of the 20th International Symposium on Artificial Life and Robotics (AROB 2015)**, Oita, Japan, January, 2015, pp. 367-372.

S. Khamphuea, T. Kondo and I. Nilkhamhang. 3D Room Reconstruction Using Depth-Map-Based Scene Complexity Analysis with Switching Strategy Guided KinectFusion, to be submitted to **Songklanakarin Journal Science and Technology (SJST)**.