Use of Consumer-grade Depth Cameras in Mobile Robot Navigation

Hao Sun

Master of Philosophy

University of York

Computer Science

September 2015

Abstract

Simultaneous Localization And Mapping (SLAM) stands as one of the core techniques used by robots for autonomous navigation. Cameras combining Red-Green-Blue (RGB) color information and depth (D) information are called RGB-D cameras or depth cameras. RGB-D cameras can provide rich information for indoor mobile robot navigation. Microsoft's Kinect device, a representative low cost RGB-D camera product, has attracted tremendous attention from researchers in recent years, for its relatively high quality of depth measurement. By analyzing the multi-data stream of both color and depth, better 3D plane detectors, local shape registration techniques can be designed to improve the quality of mobile robot navigation.

In the first part of this work, models of the Kinect's cameras and projector are established, which can be applied for calibration and characterization of the Kinect device. Experiments show both variable depth resolution and Kinect's own optical noises in depth values calculation. Based on Kinect's models and characterization, this project implements an optimized 3D matching system for SLAM, from processing of RGB-D data to further algorithms design. The developed system includes the following parts: (1) raw data preprocessing and de-noising, improving the quality of integrated environment depth maps. (2) 3D planes surfaces detection and fitting with RANSAC algorithms; also providing applications and illustrative examples about multi-scale-multi-planes detections algorithms which designed for common indoor environment. The proposed approach is validated on scene and object reconstruction. RGB-D features matching under uncertainty and noise in a large scale of data, forms the basis of future application in mobile robot navigation. Experimental results have shown that system performance improvement is valid and feasible.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the many people that helped through discussions and comments during the writing of this thesis. I would like to express my gratitude to my supervisor Dr. Nick Pears. He continuously supports my MPhil study and related research. I would also thank Chengliang Dai, Shu Cheng, Haizhou Qu, and other friends working with me together. Lastly, I would like to thank my parents and my wife for supporting me throughout all my life and during this thesis writing.

Declaration

I declare that this thesis is a presentation of original work, which I undertook at the University of York during 2011 - 2015. I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Chapter 1

Introduction

This chapter introduces Simultaneous Localization And Mapping (SLAM) and RGB-Depth cameras first. Following this, the interpretation of RGBD camera is given, which mostly concentrate on Kinect sensor, a popuilar representative RGB-D camera product. This chapter compares Microsoft's Kinect and other products, shows Kinect's limitation and advantage as vision sensor. Beyond this, the purpose of SLAM system are clarified. Challenges of using Kinect in SLAM are also specified, listed with some detailed hypothesis. At the end of this chapter, it would illustrate the whole structure of this thesis and contributions.

1.1 Background and Context

In this section, it includes basic understanding of SLAM theory and related sensor introduction, which construct the essential and preponderant hardware part of the SLAM 3D vision system. It clarified the proposed project and its challenging scenario.

1.1.1 Simultaneous localization and Mapping

For all the autonomous vehicles and mobile robots, environment recognition is one of the fundamental abilities. No matter what kind of robots or the environment, the problem of passing on information between each other is always a mission. To accomplish the typical mission like moving around and returning to the initial position, there are two problems: mapping and self-localization. In the absence of an a priori map of the environment, the robot is facing a kind of "chicken and egg problem" [26]: a map is needed to localize the robot while a pose estimate is needed to build a map. The difficulty of the problems come from unavoidable noise. Both the environment observations and pose estimations through noisy sensors lead to the accumulation of errors.

Since the two problems are intimately tied together [6], the approaches to solve the problems simultaneously are named as Simultaneous localization and Mapping (SLAM). As a technique studied for decades, SLAM not only sets the basic problems for mobile robot navigation, also becomes an essential standard to evaluate if a mobile robot is autonomous. Good SLAM solutions are executed by robots and autonomous vehicles to build up a map within an unknown environment (without a priori knowledge), while at the same time keeping track of their current location.

1.1.2 Visual Sensor and RGB-Depth Camera

With the SLAM research developed to focus on estimating the variable values of the robot pose and feature locations, feature selections and relative measures estimations become important. In 1990s, different computer vision techniques have been introduced to SLAM, offering numerous advantages. Visual sensing like stereo systems and monocular cameras provide new approaches in SLAM, by detecting the features in images and matching feature correspondences to track the landmarks in the maps.

3D visual feature detection and matching make the data association better assigned, upgrade the feature-based SLAM approaches. Visual SLAM approaches, in contrast to laser-based SLAM, focus on features and have more benefits to deal with complex environment data. As a middle level, features can be easily distinguished from environment objects level, also can be constantly observed to some extent. It seem to be an ideal engineering strategy to facilitate SLAM.

Meanwhile, kinds of sensors are developed with time to achieve the full 3D mapping information: such as laser, sonar, infra-red sensors. Early laser range finder and sonar give the robots touching capability. A new 3D sensing system with depth-scanners and visual cameras gives the robots both touching and visual recognition capabilities.

However, professional systems embarking these sensors are expensive. In recent years, the novel 3D sensing systems: RGB-Depth (RGB-D) cameras become popular. RGB-D cameras rely on either structured light patterns combined with stereo sensing or time-of-flight laser sensing to generate depth estimates that can be associated with RGB pixels [18].

1.1.3 Kinect Sensor

Until November 2010, Kinect sensor for the Microsoft XBox 360 game system was launched at the price under 120 pounds. As a highly integrated RGB-D camera device, Kinect sensor has three optical components:

1. one RGB camera;
2. one IR camera (depth sensor);
3. the infra-red projector.

It not only has the low cost advantage, but also with proper resolution and accuracy. Using Kinect as the robots' sensors, it offers 3D information which would be helpful for the landmarks recognition.

Figure 1.1 is the composition of Kinect device.

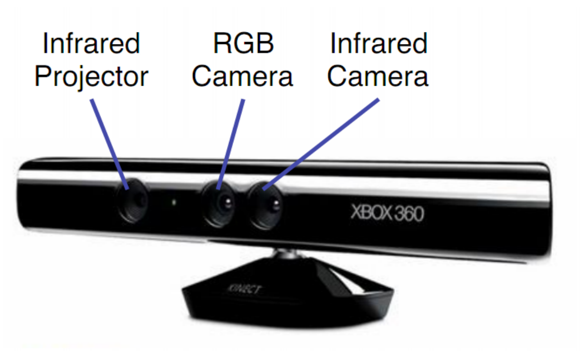

Figure 1.1: Kinect consists of Infra-red (IR) projector, IR camera and RGB camera

Here listed several RGB-D cameras products: Mesa Imaging SwissRanger 4000 (SR4000), PMDTechnologies CamCube 2.0, and PrimeSense Depth Camera (Kinect). Kinect has the low cost with proper resolution and other parameters.

Table 1.1: Main technical specification of three RGB-D cameras: Mesa Imaging Swiss-Ranger 4000 (SR4000), PMDTechnologies CamCube 2.0, and Kinect

Product Name	Technique	Range	Resolution
SR4000	ToF	5-8 meters	176 x 144 pixels
CamCube 2.0	ToF	7 meters	204 x 204 pixels
Kinect	structured light	0.6m - 4.6m at least	640 x 480 pixels

The Kinect device provides usual RGB images at a resolution of 640*480 pixels. In contrast to a ToF camera, the Kinect depth sensor outputs a 11-bit depth data at the rate of 30Hz. Kinect is a RGB-D cameras, and its depth image is produced with Primesense's

Table 1.2: Main technical specification of three RGB-D cameras: Mesa Imaging Swiss-Ranger 4000 (SR4000), PMDTechnologies CamCube 2.0, and Kinect (cont)

Product Name	Throughput	Costs
SR4000	54 fps	\$9,000
CamCube 2.0	25 fps	\$12,000
Kinect	30 fps	\$150

patented Light Coding technology. Primesense has more than 3 patents on the depth image creating process. Light Coding is newly developed around 2008 for range measurement. It is real-time and different from the time-of-flight method. As an active stereo approach, it applies projected speckle patterns and special coding/decoding techniques.

Figure 1.2 is the profile of Kinect's sensors.

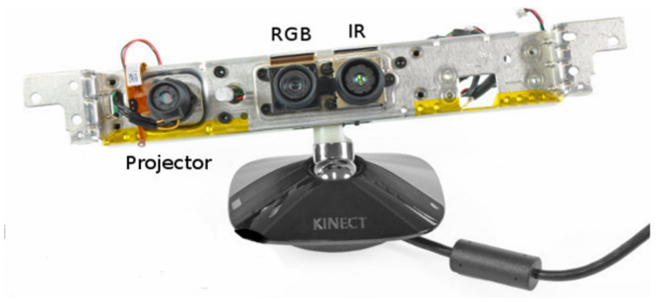

Figure 1.2: Kinect sensor profile

The IR laser projector emits a known noisy pattern of structured IR light at 830 nm. Known pseudo-random pattern of dots are captured by the IR camera and then compared to the original calibrated pattern. So the device has both the IR projector and the IR camera separated by the baseline distance. Any disturbances are known to be variations in the surface and can be detected as closer or further away. The original IR image source showed the captured IR speckle pattern projected by the infra-red projector. By measuring the disparity between a number of dots and internal calculation, the depth images are formulated. The field of view in the system is 58 degrees horizontal, 45 degrees vertical, 70 degrees diagonal.

The RGB camera shares the same 30Hz operating frequency. To enable high resolution mode, the Kinect can be switched to running at 15 frames per second (fps), which in reality is more like 10 fps at 1280x1024 pixels. The camera itself possesses an excellent set of features including automatic white balancing, black reference, flicker avoidance, color saturation, and defect correction[24].

After the Kinect device was released to market in 2010, it achieved success for the rapid market acceptance. Kinect was soon hacked for not only gaming and home entertainment applications but also other research fields. Then Microsoft company delivered the SDK source for developers. Soon the Kinect became the most popular depth sensor worldwide.

Several other RGB-D cameras products are in the same family branches with the original Kinect device. For example, Microsoft's Kinect for Windows Version1 and Version2 are based on the same technology as Kinect for Xbox. The Windows software development kit supports PC drivers and integrated with more API inside. PrimeSense Carmine 1.08, and ASUS Xtion Live device are also close relatives of Kinect. These two devices can be used the same way as Microsoft Kinect sensor. They have better RGB image quality, but does not work with USB controllers, which make them less popular than Kinects.

1.1.4 RGB-Depth SLAM

From the year of 2011, research about new Kinect device using in SLAM problem obtains more attention. Until 2014, scholars and research group have developed a new research sub-field of vision feature SLAM, called RGB-D SLAM. The main topics focus on how to apply new sensors like Kinect in SLAM solution. With new technology and new depth vision sensor coming out every year, more and more research groups start to discuss on characterization study and agile application. Indeed, in SLAM problem, sensors' selection and installation methods are essential to robot's observation. It affects the difficulty of every kinds of SLAM problem. For example, study of laser scanner pays more attention on measurements and odometry data, while vision based SLAM pays more attention on visual landmark extraction. Due to kinect's basic feature as sensor, RGB-D SLAM used to deal with indoor environment, which develop itself on both sides: data measurements and key visual features extraction and registration. Besides, improving traditional and new sensor data processing method is another topic, for instance the raw depth data processing methods and application in RGB-D SLAM sub-field.

1.2 Summary

With the choice of Kinect sensor, the RGB-Depth SLAM system project just proposes the research on these topics. The aim is to introduce and build a system with usual instances to deal with indoor environment SLAM, locally and globally improving the estimation accuracy and robustness.

Kinect's significant advantage as low cost with proper resolution, which have been discussed in Section 1.1.3. There are some other reasons leading the project to the choice of Kinect.

(1) High quality of device drivers and technique support, which make the sensor itself an reliable hardware in the whole system. Stable work are always expected in combined systems and Kinect can cooperate with various hardware models.

(2) Proper size and standard USB controllers for data uploading.

(3) Extension features recognition capabilities, such as full-body 3D motion capture, voice recognition.

Due to its design, applying Kinect sensor approach has several problems which need to be considered while assembling the measurement system. Here listed three of them:

(a) The constant wavelength can be floating in a very small scale with variations in temperature and power.

(b) The detected distance is limited by the projector strength and field of view.

(c) Some other light sources such as sunlight may interfere the performance of Kinect IR projector.

Other Kinect's characterization detail will be discussed in Chapter 3.

The main goal of this project is to improve the quality of mapping in SLAM approaches by developing better measuring accuracy and designing better matching algorithms. The content of mapping is defined as two parts: the color and the shape (depth) information from Kinect. To achieve the goal, it needs the following preparation: system calibration, multi-perspective characterization analysis and data accuracy evaluation. Building the colored point clouds in real-time is the next step. Then the key task would be algorithms designed to extract proper 3D visual features from colored point clouds, multi planes for examples. Multi-frame 3D point clouds can be compared with each other for feature matching and evaluating. Furthermore, an optimized correspondence result could be obtained by using the Iterative Closest Points (ICP) algorithm. All these hypothesis makes the estimating of pose more accurate and improve the quality of SLAM.

The remainder of this thesis is divided into the following six chapters:

Chapter 2 shows a review of the literature on SLAM problems, Kinect sensor and RGB-D SLAM related work in this field. Several tools used in the project are also reviewed in it.

Chapter 3 builds models of the Kinect's cameras and projector, standard calibration and characterization of the Kinect device.

Chapter 4 shows raw data preprocessing and refinement, 3D planes surfaces detection and fitting with RANSAC and Hough Transform algorithms, discuss on multi-scale-multi-planes problem solutions.

Chapter 5 summarises the system work in this research project. Furthermore, it concludes with experience obtained from the whole research. Finally, it presents several looking-forward suggestions for future work.

Chapter 2

Literature Review

This chapter reviews the literature relating to the research project. Firstly it presents SLAM problem in detail. This part shows the broad knowledge and mathematical modelling methods. Due to the research focus on implementation and solution, the basic theory of SLAM is mentioned only in this chapter. Following this section, studies about Kinect as an novel sensor which adopted in many research sub-field are listed. To scope the project's research topic, next section reviews from visual features extraction to cutting-edge RGB-D SLAM technique and literature in recent years. Last but not least, tools applied in this research are described as well.

2.1 Simultaneous Localization And Mapping

2.1.1 Early works on SLAM solutions

The Simultaneous Localization And Mapping (SLAM) problem was first formulated by several researchers including Jim Crowley, Peter Cheeseman and Durrant-Whyte in 1980s [52], [12]. SLAM problem is for a mobile robot in an unknown environment. It needs to build a consistent map while simultaneously navigating the robot using the map. In the research field to study the relations between robots and environment, SLAM was created as a new concept, more than normal algorithm. On the other hand, wide variety of robots lead to the situation that hardware platforms operate SLAM in many different ways. Since the SLAM problem was presented to challenge the researchers, many related research communities (Automation, Robotics, Artificial Intelligence and extra) started the pursuit to solve the problem. A successful solution of the SLAM problem can be highly valued, because it implements a method to make a robot truly autonomous.

There are several early significant developments in SLAM research. In the year of 1987, Smith and Cheesman [52] built the stochastic map, and presented a general solution for estimating uncertain spatial relationships using the map. In another article published in 1990 [51], they made accurate quantitative estimates of nominal relationships and co-variance between relative locations of objects, using a robot as example. No complex estimated moments is the only limitation.

Meanwhile, Leonard and Durrant-Whyte [12] described the geometric uncertainty for robotics, also focused on the invariant relations between geometric objects represented by geometric features. Both of two research results show the correlation between estimates of the location of objects in a map, and correlations would grow with observations.

Leonard and Durrant-Whyte [28] considered mobile robot navigation to be a problem of tracking geometric features (targets) which are present in the environment. After previous research to investigate mapping and localization separately, they offered a method and sensing strategy to solve the correlation problem with multiple sonar sensors. Compared with the early SLAM solutions (Table 2.1), Leonard and Durrant-Whyte's solution is forward-looking, also affects later development.

Table 2.1: Comparison of early works on SLAM solutions

Solution	Year	Method	Implementation
Smith, Self and Cheeseman's	1987	stochastic map, using extended Kalman filter (EKF) algorithm	simulation
Moutarlier and Chatila's [32]	1989	framework similar to stochastic map, colored and correlated noise accommodated	2D laser sensor
Leonard and Durrant-Whyte's	1991	multi-target tracking framework, consider data association uncertainty and environment dynamics	multiple sonar sensors

As seen from Table 2.1, SLAM solutions become prosperous before 1995. We make a number of typical observations:

1. Implementation is important for SLAM. When researchers consider engineering implement, the stochastic map is difficult in terms of computational complexity.

2. SLAM models needs modification. Noisy sensors will accumulate errors and affects

the results. As Leonard and Durrant-Whyte said, "To achieve genuine long-term autonomy, it is not enough just to represent uncertainty; we need to reduce uncertainty." [28] It becomes one of the main missions in later development.

3. SLAM is an open problem. Not only data association uncertainty, environment dynamics, and computational complexity are considered, more and more functionalities is introduced to add value to the SLAM solution.

4. To achieve SLAM, the hardware must be considered, for example the mobile robots platform and range measurement device (sensor).

The most common formulation is using the extended Kalman filter (EKF) to solve the SLAM problem, which was first applied by Smith and Cheeseman [51]. The basic Kalman Filter algorithm is the optimal estimator for a linear system with Gaussian noise. There are also other probabilistic approaches, such as FastSLAM algorithm solution. The EKF is simply an extension of the basic Kalman Filter algorithm to non-linear systems. The estimates made by the Kalman Filter are procedures of optimization. There are four kinds of problem which the EKF-SLAM has to face:

1. the individual landmark variances convergence.
2. computational effort.
3. data association.
4. non-linearity.

Since this project is about the improving SLAM with RGB-D Cameras, the field review is concentrated in a small region. Though the other three problems are also essential and been studied a lot, the project mainly focus on the data association problem.

2.1.2 15 years development of SLAM solutions

Since 1998, researchers have made great contributions to SLAM in many sub-fields. With better understanding of SLAM problem, the goal is becoming to develop efficient and robust SLAM solutions. In 2001, Durrant-Whyte and Csorba [11] proved that estimation errors reduces to the point where the landmarks have the precise relative locations. For more and more observation data fused, the data association problem become critical. Under certain conditions, for example a robot returns to a place and re-observe landmarks during mapping, it is difficult because the targets features are complex and represented as multi-viewpoints. It is called the loop-closure problem, shown as Figure 2.1.

The pose estimation errors accumulate after a long loop, while the robot already

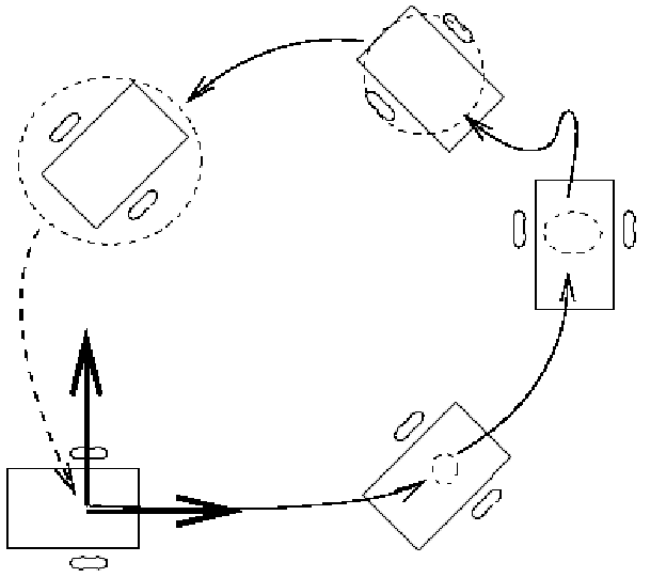

Figure 2.1: The loop-closure problem for any mobile robots. The dotted circles represent the pose estimation. The circles expand because of the accumulating errors, which lead to more uncertainty and failure of return to the start position

mapped the environment and went back to the original point. With the new associated data to remap the same location, it may cause catastrophic failure if the loop can not be detected. On the other hand, if the correct long loop is detected, it will reduce the uncertainty. The difficulty of loop-closure problem will rise while the size of the environment grows. It seems manageable for indoor environments. Since outdoor navigation tend to the GPS approaches, SLAM is always popular in the domain of indoor navigation.

Even for the indoor environment, modelling could be complex if all the real world factors are considered. The assumption that stationary environment is basic, and all the other objects' motion except the mobile robot would be treated as noise. In 1998, Thrun, Burgard and Fox [54] raised the assumption that a robot observes a series of landmarks while moving, also built map-likelihood-function to value the estimation efficiency. Their E-M Mapping could be the general method: estimate the path of the robot, given current map first, then estimate the map, given current path. Though it faces high computational costs, the model based on landmarks could be a different choice since many other researchers more depend on the range sensors.

There are two groups of methods:

1. SLAM with range-only sensors;

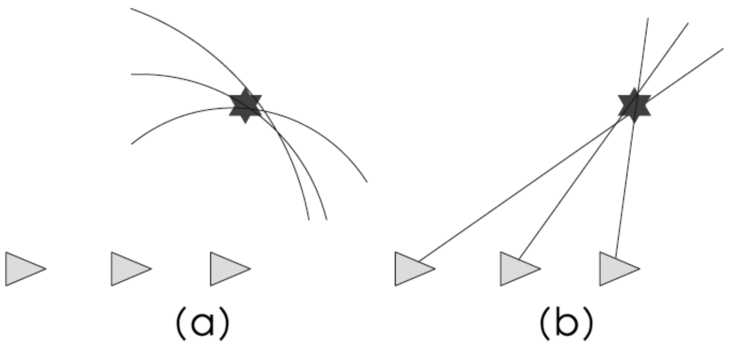

Figure 2.2: Range sensors and Bearing sensors observe a landmark location from multiple points

2. SLAM with range-bearing sensors.

The range-only sensors are two kinds: Ultra-Wide-Band devices and GPS system. They rely on a set of artificial beacons distributed throughout the environment. The more common range-bearing SLAM method uses sensors with both range and bearing. The difference is the weight. One kind of sensors focus on the range, which can measure long distance and large area, for example sonar sensors. Sonar scanning is fast but with low accuracy. Another example is laser scanning, which is accurate but slow. They usually need multiple observations to estimate a landmark location (Figure 2.2). As an alternative, sensors focusing on the bearing often use cameras and provide more informations about the environment, like high resolution images. Since cameras are cheaper than range sensors, they are widely implemented in robot platforms. Several scholars try to solve the landmark initialization problem and other problems they have to face when introducing cameras to improve SLAM solution [1], [27].

The selection of landmarks also affects the results. In early studies, the landmarks could be described simply as points, lines. In general, the mapping could be 2D or 3D. In this project, it only addresses the 3D SLAM which build a volumetric environment map. Comparing with two dimensions SLAM, three dimensions SLAM implementations need more complex feature modeling. It is almost impossible to get high resolution and accurate 3D features with only single range-bearing sensors.

With more and more studies of vision applied in SLAM, the pioneering scholars establish the foundation of a new research subject of vision-based SLAM. One famous research group was led by David Lowe. Se and Lowe use scale invariant features for mapping, and their approach can bear the long trips of the closure loop because of its independent feature recognition efficiency. [49], [48] It is the first time for the robot to recognize its location anywhere in the map without prior knowledge of its position.

Another paper as the milestone of vision-based SLAM was published by Andrew J. Davison in 2003. [8] He made an assumption that the observed rigid image feature is moving just because of the camera movement. By comparing frame-to-frame motion, the camera positions and 3D features locations estimation would be repeated to provide the real-time operation.

With the basic ideas of vision-base SLAM, all kinds of implementation come out in the following years. Marcus et al. [53] improves the accuracy of the 3D features using two cheap unsynchronized cameras and focusing on the stereo matching methods. Newman et al. [34] combine laser scanner and camera together to build the 3D laser-vision SLAM system, and obtain a 3D scan of the outdoor urban environment.

Until now, SLAM problem is still hard and not yet fully solved. However, all improvements around probabilistic methods, varied sensors and vision features tend to make the SLAM solution working better.

2.2 RGB-D Cameras and the Kinect

As mentioned above, the varied sensors for SLAM implementation improve with time. Since the vision cameras were introduced to the research of object tracking, there is a huge body of work in the area. When comparing 3D scene with 2D images, much information about the shape and geometric feature are needed to be considered. In some recent works, 3D layout or depths have been applied for improving object detection. [45], [46] The usual source to achieve 3D is monocular/binocular 2D image or a stereo video respectively. This could possibly lead to the result that 3D data can not be measuring accurately enough.

In recent years, as the novel 3D sensing systems, RGB-Depth (RGB-D) cameras is of crucial importance. RGB-D cameras rely on either structured light patterns combined with stereo sensing or time-of-flight laser sensing to generate depth estimates that can be associated with RGB pixels. RGB-D style cameras make use of both shape features and visual features for the object detection. In 2008 Workshop on Multi-camera and Multi-modal Sensor Fusion, Gould et al [16] produced a multi-model object detector with the 2D object detector and 3D information from a depth sensor. Later in 2011, Lai et al. [25] took the advantage of the new depth cameras, combined color-based and depth-based recognition together for object classification.

Rusu and Cousins presented their initiatives in the areas of point cloud perception: PCL (Point Cloud Library). The library contains algorithms for filtering, feature estimation, surface reconstruction, registration and segmentation. They used Kinect as the hardware platform expected that the Kinect and PCL working together would bring an easy 3D vision solution for most robots in the future [43].

2.3 Registration and Feature Extraction

All the modelling and calibration are inevitable preparation for creating the 3D point clouds. Comparing with 2D images, 3D point clouds are the materials for object recognition in 3D scenes. Since every detected point has its own coordinates (X,Y,Z), the collection of these unconnected 3D points is called a point cloud. Because of the lack of connectivity, it can be displayed that the points are floating in space as clouds. Normally, 3D point clouds contain position information. Considering the application of Kinect with both RGB camera and depth camera, every point also contains color information. Once the Kinect is calibrated, 3D point clouds is the most efficient and straightforward representation for the captured data. In this project, 3D point clouds are primitive data to be processed. On one hand, clusters of points are used for surface reconstruction to produce a mesh model; on the other hand, two datasets of point clouds could be aligned, known as performing registration in 3D.

Registration is a fundamental concept in computer vision, which can be defined as aligning two data sets taken from different coordinate systems. The data sets are point clouds in 3D, and the amount of raw data is much bigger than 2D images. It is important to find the rigid transformation which aligns pairs of 3D data. The process is described as looking for the translation and rotation of a target data set that produces maximum overlap with a reference data set. For this matching problem, there are two general approaches for 3D registration: point matching and feature matching.

Point matching tries to establish correspondences between two spatial points clouds directly. The unknown correspondences between the point sets make it a difficult mission. A popular approach, Iterative Closest Point (ICP) algorithm, is created to solve the problem. Besl and Mckay introduced the ICP algorithm early in 1992 [4]. In 1994, Zhang [58] proposed another ICP algorithm with improvements over the algorithm of Besl and Mckay's. The improvement on computation speed and dynamically picked maximum length strategy make the quality of registration better. The algorithm is widely used for

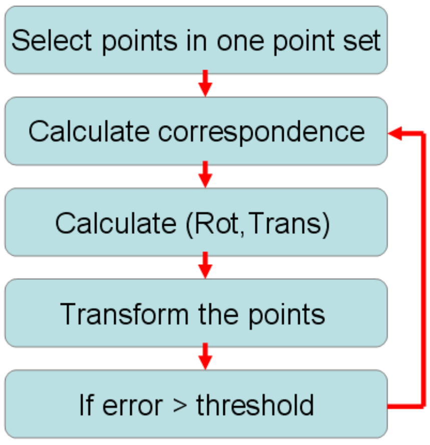

Figure 2.3: The ICP algorithm procedures. Iteratively refine transformation by repeatedly generating of corresponding points pairs

registering 3D models. ICP has become well-known because of its simplicity and easy to operate. There is one requirement that two data sets to be registered are already coarsely aligned. It means the initial estimate need to be reasonably good. In that case the algorithm would converge relatively quickly.

The ICP algorithm goes as following steps. Here C_d denotes a point cloud which is to be registered, and moved to model point cloud with a static position, C_m:

1. Select control points in the point cloud data set C_d. For all these selected points d in C_d, find the closest neighbour points m in C_m (called correspondence).

2. Calculate the optimal transformation between two point sets based on the current correspondence. The rigid transformation contains an orthogonal rotation R and a translation t. The optimal transformation minimizes the squared distances between the neighbouring pairs(enumerated with i):

3. Apply the transformation to C_d (Transform the points).

4. Repeat until the algorithm has converged or till a desired result has been obtained.

It is the naive way to perform registration, and it apparently works well in most cases. However the presented original form of ICP algorithm has several limitations. ICP algorithm requires a good initial alignment, which is a basic requirement of this approach. Since it is regarded as an optimization problem, sensitive to random noise and local minimum trap are the problems. Basic ICP algorithms converge too slow, which brings out the speed problem.

In the years of 1990s, many variants of ICP have been proposed, affecting all phases of the algorithm from the selection and matching of points to the minimization strategy [42]. Rusinkiewicz and Levoy from Stanford University write a review to summarize the class of ICP algorithms in 2001. The paper foucuses mainly on the convergence characteristics (speed, accuracy of the final answer, robustness to difficult geometry) of these ICP Variants. Rusinkiewicz and Levoy discuss the variants of ICP which affect all phases of the algorithm. They classify these variants, and evaluate their effects. Also in the paper, they proposed a combination of ICP variants optimized for high speed. The following lists summarize these variants as six stages of the ICP algorithm:

1. Selection of point. (four instances: using all points, using points from uniform sub-sampling, using random sampling, selection of points with high intensity gradient.)

2. Matching of point. (find the closest point in the other mesh, acceleration of the basic method)

3. Weighting of pairs. (constant weight, assigning lower weight with greater point-to-point distances)

4. Rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs. (rejection of corresponding points more than a given distance, rejection of worst n% of pairs based on point-to-point distance, rejection of pairs that are not consistent with neighbouring pairs, rejection of pairs containing point on boundaries)

5&6. Error metric and minimization. (sum of squared distance between corresponding points, sum of squared distance of point to plane)

The listed stages provide ways which can improve the performance of original ICP algorithms. With the development of 3D scanning techniques, more and more factors need to be considered. For example, the point clouds built from RGB-D sensors data contain the color and shape information of a target. For registration of these textured 3D data, accurate alignment of both shape and texture is required. It can be accomplished by adding a measure of color difference to the Euclidean distance metric.

Another trend is to accelerate the ICP matching speed. Closest point searching is the most time consuming step of the ICP algorithm. Various fast ICP algorithms proposed in recent years are about 3 times faster than the old ICP algorithm. Meanwhile the average mean squared error is acceptable with the high speed searching. Fast ICP algorithms also benefit from the computing hardware development, which lead to real-time registration [14].

When considering the application of point clouds registration in Simultaneous Localization and Mapping, coarsely aligned target data sets are not ensured, especially facing the loop-closure problem. In this situation, only point matching is not enough to achieve correct registration. Visual feature matching approach would assume responsibility of establishing spatial relations from the sensors data. Vision-based SLAM used to obtain feature matches between images as the robot changes position, however this 2D registration technique cannot deal with 3D motion directly. RGB-D cameras make it possible to create a dense point cloud spans the entire indoor environment. Feature matching here is about point cloud features.

Feature matching is to find correspondences between singular points, edges and surfaces. The feature-based registration algorithm seems similar to ICP algorithm in this way. It also searches for matched pairs between two input point clouds, then finds the rigid transformation which minimizes the sum of squared distance between them. Feature matching, like its name, aims to find correspondents by using detection and matching of invariant features. feature points (interest points), often have good invariant properties even viewpoint of the scene is changed. Furthermore the features have to be extracted without spending large amount of computation.

Visual feature matching algorithm calls for four steps:

1. Detect invariant features in the 3D scenes from which the point clouds obtain their shape and color information. A 3D feature detector identifies a set of 3D point clouds locations presenting rich visual information and whose spatial location is well defined.

2. Extract two sets of feature-descriptors from the point clouds and match them against each other.

3. Extract a set of matched pairs by corresponding the location of each feature to the point in the point cloud which it registers to. This is the matching procedure.

4. Find the rigid transformation which minimizes the squared distance between the matched pairs in 3D scenes.

The algorithm has similar steps as the ICP algorithm. However, it no longer require points which have already been coarsely aligned. The original dependent data source coming from Kinect sensors is fine. By the way, if a feature matching does not result in a perfect registration, the processed point clouds data can be good materials for ICP optimization for further registration. Only by combining the two matching approaches, the point clouds registration results can be better.

Interest point detection is an important processing step involved in 3D registration algorithms. In the fields of shape characterization, recognition, and retrieval, feature points provide local features that are invariant to rotation, scaling and many other transformations of 3D objects. A number of different feature detection methods have been developed.

The 2D Harris detector relies on first order derivatives of the image intensities. It is based on the second order moment matrix (or squared gradient matrix) [17]. Though some researchers developed the Harris detector in 3D forms, basically Harris detector produces an easy measure of corners, employed as a filter to order the key points [40]. It is the similar to the situation of the Hessian detector. The Hessian detector is a second order filter. The corner strength is here the negative determinant of the matrix of second order derivatives. 3D approach is partially motivated by 2D detector literatures. Instead of pixels on images, 3D approach compares each voxel with its scale space neighbors. Meanwhile the 2*2 matrix equation turns into 3*3 matrix equation. Another detector is affine-invariant versions of the previous two detectors. The affine rectification process is an iterative warping method that reduces the feature's second-order moment matrix to have identical eigenvalues.

Previous 3D feature descriptors include shape contexts [3] and spin images. The spin-image is the projection of the relative position of 3D points that lie on the surface to a 2D space where some of the 3D metric information is preserved [20]. A roughly explanation of this method: first relating the neighboring points of a feature to the normal vector of the feature, then recording three dimensional information into a 2D histogram. The problem with spin images is that information is lost in the projection to 2D. Similarly the shape contexts method describes local shape by partitioning the volume around a key point into spatial bins and then counting the number of 3D points in each bin. Both methods have some robustness to rotation and sensitive to small changes in the surface. The weakness is that neither of the transforms are invariant to scale.

Most recently developed 3D feature descriptors get their inspiration from the widely used 2D descriptor, Scale Invariant Feature Transform (SIFT) [29]. The SIFT descriptor is a 3D histogram of image intensity gradients, made to be invariant to image scaling, rotation and change in illumination and 3D camera viewpoint. The features are highly distinctive, effectively reducing the probability of disruption by occlusion, clutter or noise.

Local feature descriptors depends on the choice of key points locations. It is not hard to locate suitable key points in 2D images, for example the corner points with intensely change in gradient. In 3D point clouds, the key points need to satisfy more complicated condition.

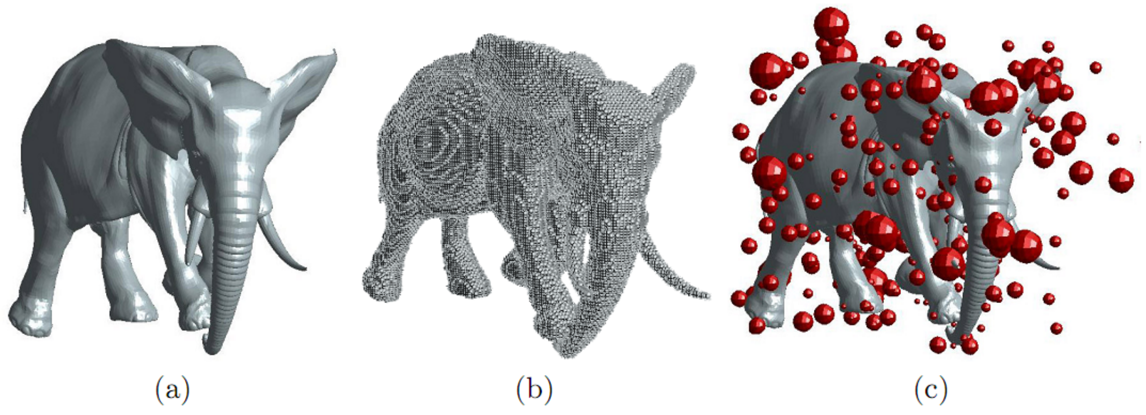

Figure 2.4: Illustration of the detection of 3D SURF features. The shape (a) is voxelized into the cube grid (b). 3D SURF features are detected and back- projected to the shape (c). where detected features are represented as spheres and with the radius illustrating the feature scale [22]

It requires that surfaces about key points have high spatial gradients in all the three directions. 2D SIFT's success owes to the image gradient orientations as basic descriptive element. Its 3D generalizations follow this idea and choose the principal direction or surface normal direction where surfaces intensity change with highest speed [15].

Speeded Up Robust Feature (SURF) is another robust local feature detector partly inspired by the SIFT descriptor [2]. As a solution to detect and describe invariant features in input scenes, it is easier to use in implementations because it is available in OpenCV. In 2010, Knopp et al. [22] represent their innovative local descriptor as a 3D extension to SURF. The extraction of the 3D features and combination with the probabilistic Hough voting framework works well for 3D shape class recognition. It performs faster than SIFT and gives an extra option to the user. One weakness of SURF is that it dose not support color images and 3D scenes. The point clouds can be supplied to the algorithm in gray scale.

Another representation of new feature matching algorithms was developed by Mian et al [30]. This tensor base representation performs by mapping the 3D points onto the 2D retinal plane of the sensor and performing a 2D Delaunay triangulation over the mapped points. After triangulation, the points are mapped back to the 3D space. Unlike the spin images to reduce the dimensionality and describe surfaces in 2D, it creates a 3D tensor by recording a 3D histogram of intersecting surface areas. Due to coordinate system instabilities, the method can not describe rotation invariant features along all three dimensions. In further development it tends to create a descriptor with both tensor representation basis and rotation invariance technique from SIFT.

Once the feature description process has been completed for two point clouds, correspondence by matching feature descriptors from each data set are discovered. A series of correspondences represented by feature pairs can help to compute a transformation, then to align the features of the two data sets. Not only several feature description algorithms can be combined together for the alignment, but also the point matching ICP algorithm carries on the fine registration to eliminate error and optimize results.

2.4 Using the Kinect for SLAM

Many groups studied the RGB-D data to build 3D point clouds for different purposes. Some of them employ the data for object classification, labeling and augmented reality [24]. Some of the groups just conduct Kinect as an easy human-computer interaction method for education and business presentations. Of course, there are a large part of research work relating the robot SLAM problems.

The Kinect could improve a SLAM algorithm in these ways:

1. With the depth information in every RGB pixel, it is no longer necessary to make a guess on the distance of an observed feature;

2. The estimation of feature positions can achieve a more accurate estimate because of direct 3D measurement;

3. It creates entire dense point cloud moving with the robots, not just sparse features.

There are many ways to use the Kinect for SLAM. There are surveys for several of them in this section.

–RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments [18]

This paper presents the general 3D mapping procedure for indoor environments using Kinect. It shows that RGB-D camera can capture rich information such as usual sparse feature and dense point clouds. The algorithm uses rich visual features for the frame-to-frame alignment.The core framework is a novel ICP variant called RGBD-ICP, which not only extract sparse visual features, also associate features with their depth values to generate feature points in 3D. If the SIFT visual features are presented enough, the ICP loop would not match frames with pose information. If visual features are not

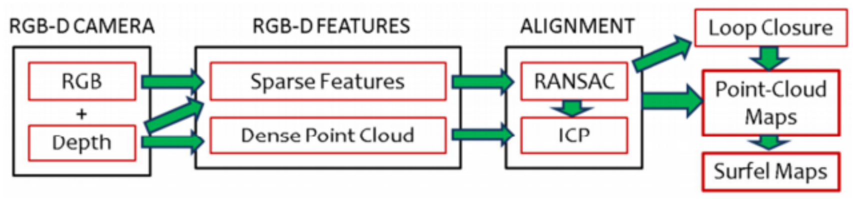

Figure 2.5: Overview of RGB-D Mapping. The algorithm uses both sparse visual features and dense point clouds for frame-to-frame alignment and loop closure detection.

enough, point-to-plane ICP procedure would generate more accurate alignments. There is a weighting value between the SIFT and dense point components. The implementation extracts features in 150 ms, runs RANSAC in 80 ms, and runs dense ICP in an average of 500 ms. The paper brings a novel algorithm for frame matching and loop closure detection with integration of depth and color information. It proves robot navigation with rich 3D maps can be built with inexpensive RGB-D cameras. Besides, the paper points out their shortcomings that the RGB-D mapping implementation is not real-time. It shows the limits of RGB-D cameras: small field of view and less depth precision, but doesn't give solutions to improve the basic performance of Kinect device. After all, it is the first paper talking about Kinect application in SLAM.

–Real-time 3D visual SLAM with a hand-held RGB-D camera [13]

The Germany research groups brought out their RGB-D SLAM system in 2011. The four steps procedure is similar to the last paper. The difference is that they applies SURF features from the color images instead of SIFT features. The depth images are evaluated to obtain the 3D correspondences between frames, and lead to the pose refinement. The ICP algorithm and further pose graph optimization would output a colored point cloud. The experiments are moving Kinect slowly around the target object and acquiring about 12 RGB-D frames. The advantage of their approach is that it is totally open source and supported by the ROS (Robot Operating System) organization. However, it also shows two weakness: 1. The hand-held camera doesn't have to solve the problem of automatic view point selection; 2. They haven't evaluate the system with ground truth information. If put the two papers together, it shows the basic process of how to use Kinect as sensors to improve SLAM solutions.

–KinectFusion: Real-Time Dense Surface Mapping and Tracking [33]

This paper was firstly presented in the ACM SIGGRAPH 2011 Talk. The work was performed at Microsoft Research. Their detailed method builds 'the first system permits

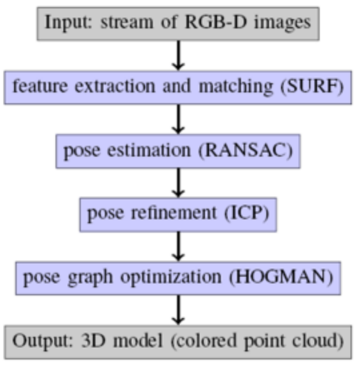
Input: stream of RGB-D images
feature extraction and matching (SURF)
pose estimation (RANSAC)
pose refinement (ICP)
pose graph optimization (HOGMAN)
Output: 3D model (colored point cloud)

Figure 2.6: The four processing steps of ROS approach

real-time, dense volumetric reconstruction of complex room-sized scenes' using a hand-hold Kinect device. As the successor of the real-time SLAM with a monocular camera in small workspaces, the KinectFusion method is more powerful with both monocular camera and dense optical flow matching for depth features mapping. The attracting part of the method is its system work-flow. When processing of depth measurement, pose estimation and reconstruction is repeated, the result of 3D point clouds is not just simple matching together. The alignment between looping closure frames is clearly better and reconstruction artefacts reduced. One of the key concepts in the real-time tracking and mapping system, is fast surface fusion and accurate tracking of the camera pose. The parallel algorithms and the GPU hardware work together to create the real-time GPU based ICP, which extract great advantage for the system. The idea and implementation of this paper is cutting-edge. It shows the Kinect's potential in mapping for medium and small sized room. The mapping of large scale area facing some additional challenges which is not yet solved.

–Range Sensor Based Model Construction by Sparse Surface Adjustment [41]

Improving SLAM solution can be achieved by reducing inconsistencies and the overall uncertainty in maps. This novel approach focuses on the key points of reconstruction of environments or objects with range data. Under the assumption that each range measurement is rigid, the graph structure can be refined by recomputing the data associations. The object surface model and RGB-D cameras model are the core conceptions for the improving, which are demonstrated by several experimental results. This paper provides some subtle designs for the modeling of given range data and also help to improve the SLAM.

–Realtime Visual and Point Cloud SLAM [14]

The 30Hz high frame rate of Kinect devices is not fully exploited. In 2011 Fioraio and Konolige presented their technique for fast registration. The algorithm for Visual-based SLAM perform real time and generalized ICP on dense range images. Overall global alignment using both depth and visual matching features in a uniform framework. The experiment in this paper proves ICP and visual feature matching fail separately, and only the combination of the two leads to a correct solution. Besides, the ICP method partly solves the depth data noisy problem for Kinect sensor.

–Adaptive Data Confidence Using Cyclical Gaits On A Modular Snake Robot [31]

This paper is giving an example about using Kinect in an implementation of 6-DOF SLAM. Its constructive ideas of how to setup the RGB-D SLAM code package in a robot operating system is helpful. The method it uses is the same SURF feature detector which talked before in [28]. However, the experiments lead to some new concepts which need to be considered. For example the motion design of robot with Kinect as its sensor, and the hardware platform preferences for the SLAM mission.

–Using Depth in Visual Simultaneous Localization and Mapping [47]

In recent years, visual-based SLAM task can be divided into two parts: estimating transforms between frames and optimizing the pose graphs. Scherer et al presented their novel approach which doesn't rely on the measurement of all pixel positions in the depth images. Since depth measurements from the very first RGB-D frame are used to initialize the map at the correct scale, the error also accumulates if only using the depth measurements. in order to minimize the re-projection error, the paper presents a method to optimize both camera poses and 3D points. Specially, by adding depth constraints to the bundle adjustment, the SLAM accuracy could be improved. The authors made a rigorous study about the depth errors modelling, and raised the convinced results and solutions. The Kinect is useful in SLAM improving, while there are still a number of hard problems. Using RGB-D cameras in SLAM is not just deploying a sensor, but also making the device work more reliably for the robot navigation.

Besides the above research work, there are some other applications. For example, Nicolas's RGBDemo, contribute a lot to the calibration and visualization of Kinect output. The projects of MRPT(Mobile Robot Programming Toolkit), also provide open-source technical support for the Kinect developers.

2.5 State of Art

Looking over Kinect related works between 2011 and 2014, these methods with leading progress are mostly basic combination of popular vision algorithms and this new sensor. They have the advantages of quick engineering implementation and 2D to 3D data processing skills, which makes researchers easy to understand and applied into more wider areas. In SLAM research field, community groups produce and improve these state-of-art systems over years and years:

1. MonoSLAM
2. PTAM (The Parallel Tracking and Mapping algorithm)
3. FAB-MAP [7]
4. DTAM (Dense tracking and mapping in real-time)
5. KinectFusion [33]

Gaps in current knowledge are following:

1. Kinect camera calibration and metrics characters are needed to improve the modelling for following algorithms results.

2. In different types of environments, make improvement of the algorithms' robustness and adaptability

The research focuses on these two areas: modelling and flexible algorithms. The target is to find state of art algorithms/methods in particular defined environment, with properly calibrated Kinect cameras. Dealing with modelling and algorithms is an exploration and exploitation procedure. The following chapters separately discuss two faces of this dilemma, also show understanding of the optimised choices between proper modelling and developing algorithms.

2.6 Summary

The chapter reviews literature of SLAM problem theory, Kinect sensors, RGB-D SLAM systems and visual features extraction methods.

Chapter 3

Kinect Calibration and Characterization

In this chapter it shows modelling of Kinect cameras first. Secondly, standard stereo cameras calibration method for Kinect is presented. Debate about Kinect's characterization and errors is displayed following the calibration. With the results and analysis, integrated de-noising and filters are listed at last.

3.1 Stereo Cameras Calibration

All of the IR camera, RGB camera, and IR projector need calibration, in order to create 3D point cloud correctly. Actually, the RGB camera and IR camera can not be highly trust in SLAM research unless they are calibrated. The RGB camera follows the restricted single view geometry and standard calibration procedure, to obtain its intrinsics parameters. Meanwhile the pair of IR camera and IR projector perform as similar system of stereo cameras. The project starts with building the camera and projector models, applying the most specialized and simplest camera model: the basic pinhole model, which is shown in Figure 3.1 .

The transformation from the physical 3-dimensional world with coordinates (X,Y,Z) to the points in the image plane with coordinates (x,y) should be classified as central projective transform. Any defined point P in real space has 3D coordinates $\mathrm{P} = (X, Y, Z)^T$, projected onto an image plane and represented by a point $\mathrm{x} = (x, y)^T$. The homogeneous coordinates are given as $(x, y, 1)^T$. From properties of these similar triangles,

$$x = \frac{f_x X}{Z}, \; y = \frac{f_y Y}{Z} \tag{3.1}$$

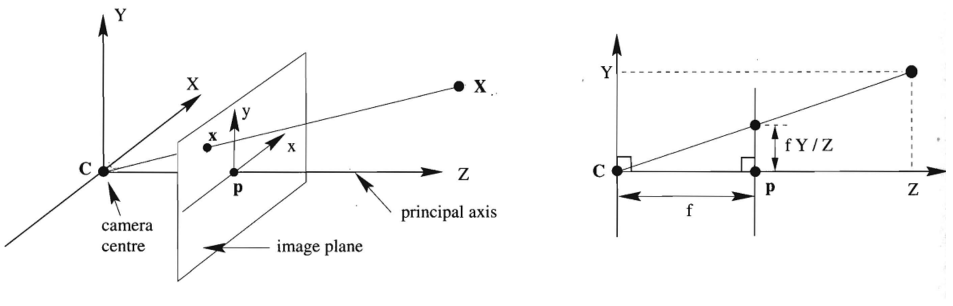

Figure 3.1: Basic camera pinhole model

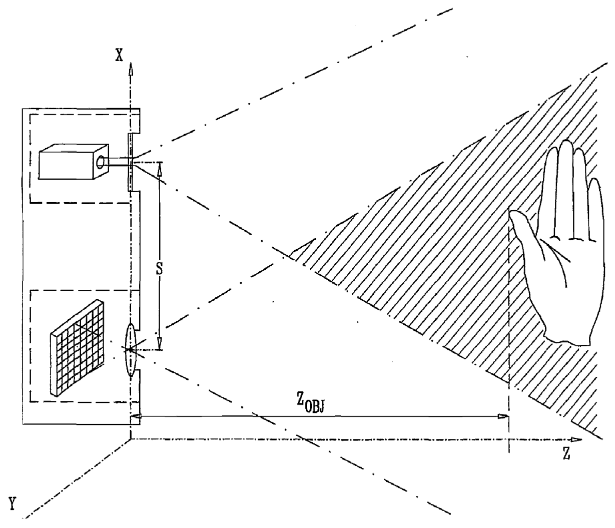

Figure 3.2: Specific camera models for the Kinect sensor. The pair of IR projector and IR camera combine a system of stereo cameras. An object's depth can be detected with the calibrated system

At the IR camera image plane

$$\overrightarrow{x}_{ir} = \mathrm{M}_{ir}\mathrm{X} \tag{3.2}$$

Here M_{ir} is named camera projection matrix, contains the rotation and translation value.

$$\mathrm{M}_{ir} = \mathrm{K_{ir}}[\mathrm{I}|\mathrm{O}] \tag{3.3}$$

Similarly at the projector plane,

$$\overrightarrow{x}_{p} = \mathrm{M_p X} = \mathrm{K_p}[\mathrm{R}|\mathrm{t}] \tag{3.4}$$

Since the projector and camera are strictly parallel (which testified by experiments), I, R both equal to identity matrices. While the distance between the projector and IR camera is defined as S,

$$\mathrm{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \mathrm{O} = \begin{bmatrix} S \\ 0 \\ 0 \end{bmatrix} \tag{3.5}$$

According to the assumed camera model, IR camera intrinsic parameter matrix

$$\mathrm{K_{ir}} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Because of the lack of information about the projector, here it is assumed the projector and camera have the same focal length f and optical center. It means both the image planes are at the same XY plane.

$$\mathrm{K_{ir}} = \mathrm{K_p} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$\overrightarrow{x_{ir}} = \mathrm{K_{ir}}[\mathrm{I}|\mathrm{O}]\mathrm{X} = \begin{bmatrix} f_x & 0 & x_0 & S \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x(X+S) + x_0 Z \\ f_y Y + y_0 Z \\ Z \end{bmatrix} \tag{3.8}$$

$$\overrightarrow{x_{p}} = \mathrm{K_p}[\mathrm{R}|\mathrm{t}]\mathrm{X} = \begin{bmatrix} f_x & 0 & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x X + x_0 Z \\ f_y Y + y_0 Z \\ Z \end{bmatrix} \tag{3.9}$$

Dividing by Z, then based on the assumption the projector image plane shares the same focal distance as the IR image plane, disparity (in pixels) between projector and camera can be defined:

$$\mathrm{d} = \begin{bmatrix} \frac{f_x S}{Z} \\ 0 \\ 0 \end{bmatrix}, \quad Z = \frac{f_x S}{d} \tag{3.10}$$

For each real world point at different depth distance, there will be a corresponded disparity at the x-axis of the image plane. X and Y are easy to calculate with the Z-coordinate value. For IR camera,

$$X = \frac{(x_{ir} - x_0)Z}{f_x} - S, \quad Y = \frac{(y_{ir} - y_0)Z}{f_y} \tag{3.11}$$

The disparity is always the key point of all the depth sensor procedure. In some way, this ideal camera model can be recreated to investigate the phenomena of structured infrared light and the procedure to measure depth. The main motivation behind the techniques is to acquire a depth map. For two corresponding images, comparing the disparity of every point on the image can create such a map. If the reference images are fixed and accurate in real world, all other images only have to consider one auxiliary variable quantity:

$$\Delta = d_{obj} - d_{ref} = \frac{f_x S}{Z_{obj}} - \frac{f_x S}{Z_{ref}} \tag{3.12}$$

Then the depth could be captured by measuring in pixels on image.

The Kinect has an image reference of what the pattern looks like from the cameras viewpoint, when all points in the surface refer to a certain, known distance. This is based on the Primesense's Patent 'Depth Mapping Using Projected Patterns'. By comparing the horizontal position of a point in the captured image to its corresponding horizontal position in the reference image, a binocular disparity can be extracted, which in turn can lead to the depth of the pixel by calculation.

Kinect only offers the disparity value as 11-bit number: 0-2047. It creates the 640x480 array. The Kinect itself actually does not calculate the depth, so the formulas between the disparity value and the real world depth distance should be estimated or established.

$$\Delta = \frac{f_x S}{Z_{obj}} - \frac{f_x S}{Z_{ref}}, \quad Z_{obj} = \frac{f_x S Z_{ref}}{f_x S + Z_{ref}\Delta} = \frac{Z_{ref}}{1 + \frac{Z_{ref}}{f_x S}\Delta} \tag{3.13}$$

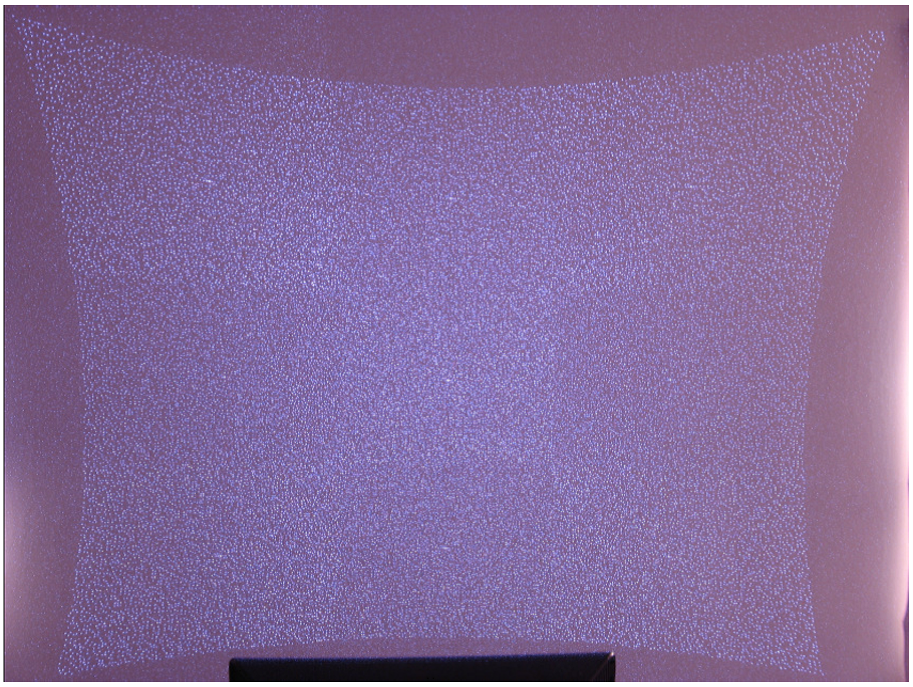

Figure 3.3: Known pseudo-random pattern of dots, corresponded disparity appears while the object's moving in Z-axis, changing the depth distance to the Kinect

The relation between the disparity value Δ and the depth z_{obj} is modelled using the equation:

$$z_{obj} = \frac{1}{\alpha(\Delta - \beta)} \tag{3.14}$$

where α and β are part of the depth camera intrinsic parameters to be calibrated.

Since the Kinect's first release in 2010, this low-cost depth camera has been fully studied. The field of calibration is pushed forward by practitioners, working on real-world engineering problems. As one of the core technologies, depth detecting mechanism attracts lots of research groups' attention. By modelling the cameras and calibrating the parameters, it is easier to understand how the depth detecting system works. The basic mathematical model mentioned above is used by most groups. However, there are different methods dealing with the model for calibration.

Smisek et al. presented their Kinect geometrical model in 2011 [50]. It only gives the depth distance results without proving, but the simple model focuses on the application, and associates the geometry of both IR camera and RGB camera, which is more helpful for data processing and registration of original images. When talking about calibration, the estimated distortions effects of both Kinect cameras are considered in order to increase the accuracy of 3D measurement.

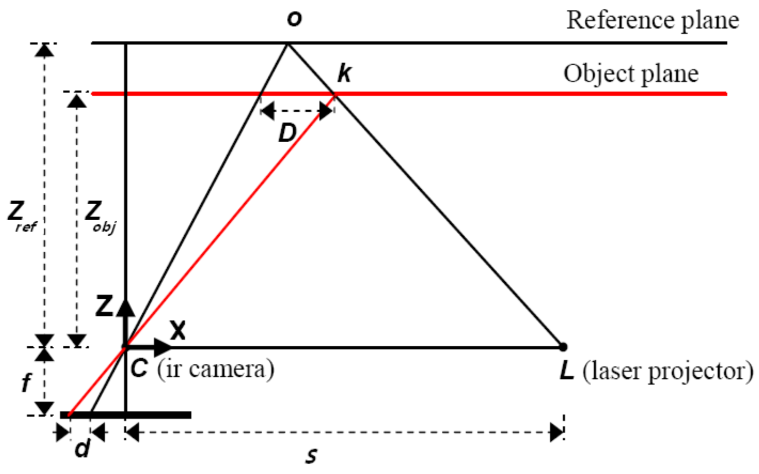

Figure 3.4: Schematic representation of depth-disparity relation

In the modeling part it introduces the disparity d in Equation 3.10. This disparity is presented in ROS's technical website written by Konolige and Mihelich [23]. They firstly point out Kinect uses disparity offset in pixels from a calibrated reference image to evaluate infrared image. It is the way to transform the structured light pattern into depth value.

In 2012, Khoshelham brought out an novel investigation of the geometric quality of depth data obtained by the Kinect sensor [21]. The similar mathematical model in his article is directly created with the assumption of reference plane. The mathematical model equation

$$Z_{obj} = \frac{f_x S Z_{ref}}{f_x S + Z_{ref} d} = \frac{Z_{ref}}{1 + \frac{Z_{ref}}{f_x S} d} \tag{3.15}$$

can be explained by Figure 3.4. Furthermore, author listed the the calibration parameters involved in this mathematical model. Besides the parameters from standard camera calibration, the base line length S and distance to the reference plane Z_{obj} also need calibration. This observation hits the difficulty of Kinect calibration but with no more formulation.

3.2 Characterization of Kinect

Recognised information of cameras calibration parameters is essential for generating accurate coloured point cloud. Only after calibration, cameras can be adapt to real world measurements. The first part of contribution is about Kinect's cameras calibration. Based on the theory of camera calibration and 3D reconstruction, the project chooses a procedure

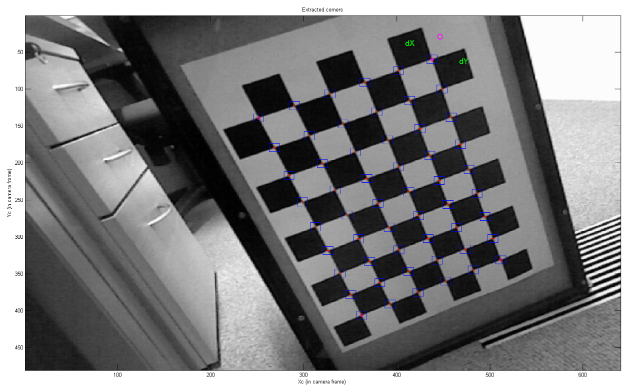

Figure 3.5: Click the corner points on chessboard grid and calibrate the RGB camera automatically

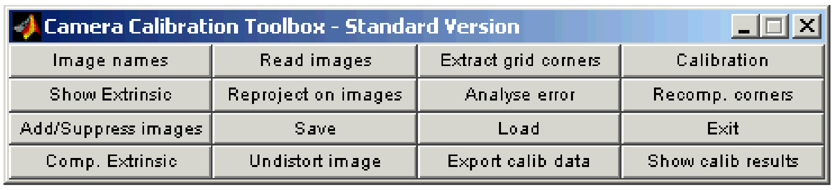

Figure 3.6: Interface of Matlab Camera Calibration Toolbox

to simply calibrate the Kinect's RGB camera with practicality. It has extended research line about chessboard pattern calibration using 'Matlab Camera Calibration Toolbox'. Calibration object is a chessboard, shown in Figure 3.6.

There are some reasonable benefits with chessboard pattern. It is a plane with corner points easily to identify and extract uniquely. When the Kinect is fixed, the chessboard is held in various poses in front of the camera, trying to cover as much of camera's field of view with these different poses. Corner points of the calibration object are determined up to sub-pixel accuracy using an automatic corner feature detector. Pictures taken in different orientation will prevent a bad estimate of the various parameters.

Instead of OpenCV calibration functions, this project applies Matlab Camera Calibration Toolbox, because its procedures satisfy the challenge with practicality. Followed with each step to the end: loading calibration images, extracting image corners, running the main calibration engine, displaying the results, the intrinsic parameters data of RGB camera showes as:

1. Focal length: fc = [550.307202370897240 ; 548.427979065932850];
2. Principal point: cc = [325.009050692525650 ; 255.963736813278590];

The focal length in pixels is stored in the 2x1 vector fc, which results from the implementation of the famous camera calibration method [59].

The principal point cc is often hard to derived directly and estimated reliably. It is known to be one of the most difficult part of the native perspective projection model to estimate (ignoring lens distortions). In this case, it is sometimes better (and recommended) to set the principal point at the center of the image (cc = [(nx-1)/2;(ny-1)/2]) and not estimate it further.

3. Skew coefficient: alpha_c = 0.000000000000000 (unit in degrees);

4. Distortion coefficients: kc = [0.198605423329255 ; -0.459285830195762 ; -0.008656507034746 ; 0.001113169340110 ; 0.000000000000000];

Image size: nx = 640; ny = 480; From the CMOS sensor size and 640*480 resolution frame, derived pixel size equals 9.3μm.

Table 3.1: Color camera internals

Color internals							
f_{cx}	f_{cy}	u_{c0}	v_{c0}	k_1	k_2	k_3	k_4
550.30720	548.42798	325.00905	255.963737	0.198605	-0.45929	-0.00866	0.00111

Nicolas Burrus [39] provide the software toolbox (Kinect RGBDemo v0.6.1.), which doesn't grab IR images, only RGB images and RGB cameras calibration file. Applying this method , factory calibration results of the same kinect are saved in ROS calibration files: calibration.yml. Here showes the Intrinsic matrix and distortion coefficients for Kinect RGB camera:

1. RGB_camera_matrix: $\begin{bmatrix} 528.0144043 & 0.00000000 & 320 \\ 0.00000000 & 528.0144043 & 267 \\ 0.00000000 & 0.00000000 & 1.00000000 \end{bmatrix}$

2. RGB_camera_distortion_coefficients: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

3. shift_offset: 1090

4. projector_depth_baseline: 0.07500

These parameters are written from the original calibration of Kinects.

This project also performs demonstrating RGBDemo to refine these original calibration. Actually, it is not the unique method to manually calibrate if applying OpenNI. An alternative line of calibration research also has extended. It has pretty good default parameter to align to color and depth images. OpenNI comes with a predefined calibration stored in the firmware that can directly output aligned depth and color images with a virtual constant focal length. Most applications will be happy with this calibration and require no more any additional steps. However, some computer vision applications such as robotics might require a more accurate calibration.

The factory calibration results are not that accurate enough in these scenarios. They are stored onboard, managed by OpenNI for images un-distortion, and for registering the depth images (taken by the IR camera) to the RGB images. Therefore, the depth images in datasets are re-projected into the frame of the color camera, which means that there is a 1:1 correspondence between pixels in the depth map and the color image. The function in OpenNI would operate the alignment to produce both images from the perspective of the RGB camera.

Considering the cameras modelling part above, IR camera's calibration can be accomplished as the same way theoretically. If executing the same camera calibration, it needs to capture images of a chessboard pattern from IR camera like RGB camera. When capturing images from the IR camera, a creative thinking is to block the projector for good corner detection in chessboard images. Otherwise, the corner detection may fail. Here are two figures of the IR images : One is the original with dot patterns; the other is after the blocking of projector.

When calibrating the images with no emitting light from projector, the remaining difficulty with the objective comes from not enough lighting sources in environment. If there is a light to emit IR rays, the result will be better. Matlab Camera Calibration Toolbox has the ability to deal with light intensity enhanced images. Considering about the scale factor due to image capture problem, the refined results in this way of calibration is shown below:

1. Focal Length: fc = [593.58503 589.29362] [3.30435 3.17770]

2. Principal point: cc = [321.52359 238.12760] [4.04813 4.56828]

3. Skew: alpha_c = [0.00000] [0.00000] = angle of pixel axes = 90.00000 0.00000 degrees

4. Distortion: kc = [-0.11740 0.33334 -0.00222 -0.00007 0.00000] [0.01311 0.05507

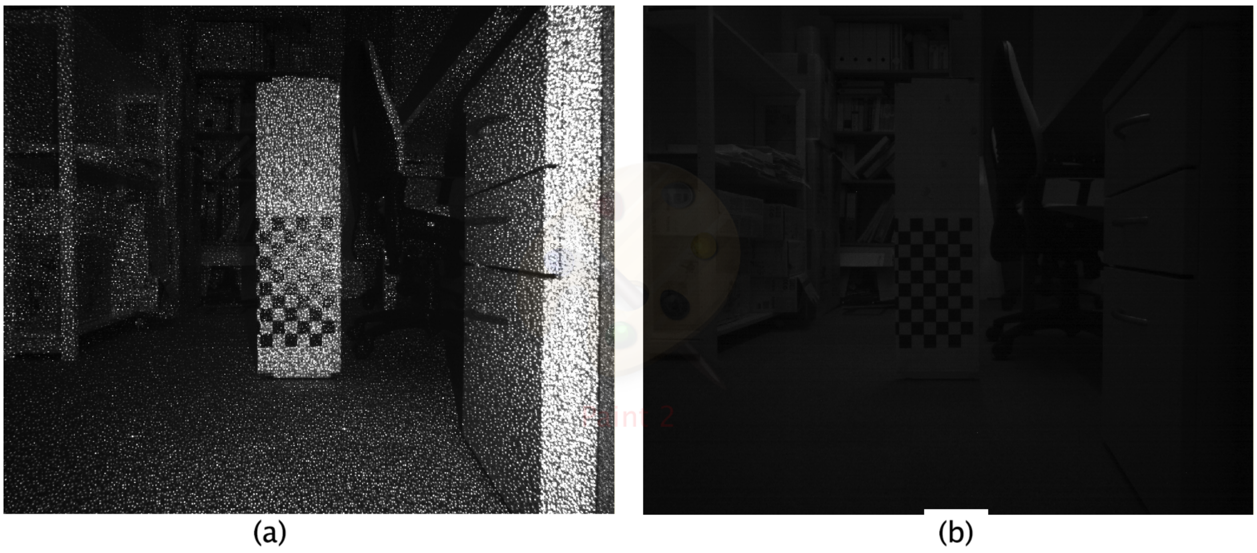

Figure 3.7: Calibration of the IR camera: (a). chessboard pattern images with dot patterns and cannot finish corner detection; (b). after blocking the projector, capture the images for corner detection and calibration

0.00158 0.00141 0.00000]

5. Pixel error: err = [0.56907 0.46819]

It showes that pixel error is less than one pixel. Similar results can be derived from repeating experiments, as proposed calibration scheme in this project.

Table 3.2: IR camera internals

IR internals							
f_{ix}	f_{iy}	u_{i0}	v_{i0}	k_1	k_2	k_3	k_4
593.58503	589.29362	321.52359	238.12760	-0.11740	0.33334	-0.00222	-0.00007

After calibration, all the intrinsic camera matrices of IR camera, RGB camera, and distortion parameters are catalogued. Keeping the chessboard constant and capture both RGB and IR images together repeatedly, extrinsic parameters as relative R and T Matrix can be computed:

1. relative_R_matrix: $\begin{bmatrix} 0.99999985 & 0.00048633 & 0.00008552 \\ 0.00047829 & 0.99999986 & 0.00008296 \\ 0.00007645 & 0.00008311 & 0.99999941 \end{bmatrix}$

2. relative_T_matrix: $\begin{bmatrix} 0.00061669 \\ 0.00053755 \\ 0.00021411 \end{bmatrix}$

3.2.1 Depth Data and Disparity

The following part debates on depth data and disparity. In Equation 3.14, it describes the depth distance of target, which is only related with disparity and intrinsic parameters. The relation between the disparity value Δ and the depth z_d is modelled by performing the equation:

$$z_d = \frac{1}{\alpha(\Delta - \beta)} \tag{3.16}$$

Where α and β are part of the depth camera intrinsic parameters to be calibrated.

As introduced in Chapter 2, Kinect sensor only outputs 11-bit depth data, counted as 0-2047 different values. The values are not distributed uniformly. There are more values from 50cm to 200cm, less values out of 5 meters. Of course, Kinect's depth measurement as a function of distance does not scale to be linear. From original disparity data, it asks for a formulas estimation to transform these discrete values to real world depth values.

Here list three modelling results of the estimated formulas, which need to contrast and analyse both empirically and theoretically:

1). OpenNI provides the Function Formulas in "Depth Generator".

2). RGBDemo's operation result: Depth calibration was determined experimentally, by measuring the reading of the center pixel in the depth image, and dealing a regression on the data. From their data, a basic first order approximation for converting the raw 11-bit disparity value to a depth value in centimeters is:

$$z_d = \frac{100}{-0.0030711016 * rD + 3.3309495161} \tag{3.17}$$

Here rD represents raw disparity, equals Δ. This approximation of centre pixel is approximately 10 cm off at 4 m away, and less than 2 cm off within 2.5 m.

3). A better approximation is given by Stephane Magnenat (Open Kinect Google Group) in meters:

$$Z_d = 0.1236 * tan(rD/2842.5 + 1.1863) \tag{3.18}$$

Adding a final offset term of -0.037 centers the original data. The approximation has a sum squared difference of 0.33 cm while the 1/x approximation is about 1.7 cm in the centre pixel.

In this project it implements RGBDemo function to transform raw disparity data to real world depth. By computing the depth-value volume of 640*480 matrix data (saved as YML file, independent with RGB volumes), the depth images after calibration can be achieved, seen as Figure 3.11. It is displayed as gray scale, which is different from usual Kinect's depth images used in entertainment. Kinect's depth images applied in entertainment performance is achieved by Microsoft's SDK, only calibrated after manufactured (factory calibration). However, since the formulas is given as experiments, it requires assess effectiveness of whole concepts. Formulas can be modified with this project's results in further experiments and data sets.

The statistics of calibration results are detailed in Section 3.2.2. Only with these repeatable results, the method demonstrated in this part can be assessed to be reliable.

After all these modelling, calibration, and transforming sensor data to measure real world depth, the parameters to create 3D point cloud are ready. This procedure is called modelling and data pre-processing. During it, the errors come mainly from following aspects:

1). Optical system modeling errors, which shown as camera intrinsic and internal values calibrated;

2). Calibration method system errors, which explained as bias and variance, using standard deviation of relative error, a method for evaluation of uncertainty. Calibration accuracy versus depth camera noise level, versus number of model planes, and versus correct noise model.

3). Disparity value to real depth modelling errors, as Equation 3.12. When Δ is smaller, it means Kinect is more accurate. It derives a conclusion that Kinect's accuracy is a quadratic function related to the distance. The result is also shown in following Kinect characterization section.

Except Kinect device's system error, these errors added while our procedure of creating the 3D point cloud should be reduced with further methods, such as refined modelling with factors talked in Primesense's patent (Depth-varying light fields for three dimensional sensing). More reliable calibration, and subdivided fitting curve for disparity data transforming are needed.

3.2.2 Characterization of Kinect system

In this sub-section, It discusses the work of data understanding of the created 3D point clouds. First, it chooses a flat plane as the most simple target object. The white board is chosen as experiment object, shown in Figure 3.8.

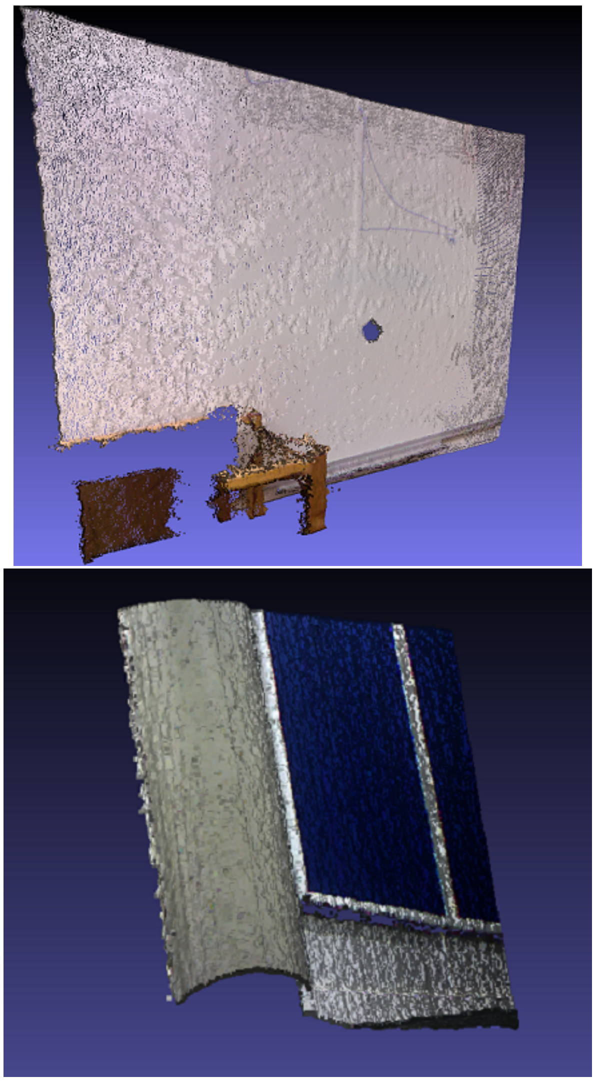

Figure 3.8: Choosing flat whiteboard and curved pillar as target object for error analysis

The series of experiments are designed to examine Kinect's performance at different distance to a flat board: Capturing 100 frames data at fixed distances to white board. Since Kinect's independent performance in different distance, there should be at least 8 series: at 50cm, 100cm, 150cm, 200cm, 250cm, 300cm, 400cm, 500cm. It also contains 2 special series like 40cm and 800cm.

After data capturing, the next step is to calculate random errors at chosen position pixels. In a small group test of only 15 images, the mean-square deviation is 0.00184. That's an example of the whole experiments.

In order to make the experiments reliable, the pixels choosing in images would be agglomerated in four small size rectangles (80*60), at left-top, right-top, left-bottom and right-bottom positions of the whole images. To evaluate the system errors during 3D point clouds generation, the rectangles part organised in point clouds will compare with ground truth–real ones measured and drawn on images.

During the experiments, there are some other characterization of Kinect which are observed. One kind of them is about pixels, such as 'Flying pixels', or CDT (common distance transform). It appears especially at object boundaries. Others are low pixel resolution, individual pixels depth feasible measurement problem.

On the other hand, the rigid model of IR camera and IR projector effect some of the characterization of the Kinect system. Here is one example of the shadow effect. As Figure 3.11 shows, none of IR dots could ever reach the objects behind the obstacle. They're stuck in the closer obstacle's IR shadow. Since the Kinect can't see through or around objects, where the unobserved side of the object is hallucinated. There will always be blanks of the scene that are occluded or blocked from view without any depth data. The IR camera is in a small distance, however be able to show the shadow on IR image. This phenomenon is called occlusion. Which parts of the scene will be occluded is determined by the position and angle of the Kinect relative to the objects in the scene.

Following section demonstrates another experiments achievements: object as the pillar in an indoor environment, like corridor of the building. This clusters of experiments are designed to detect curves in 3D point clouds. It is similar to the last series of experiments, however has different ground truth measurement. Instead of plane fitting, curve surface fitting with 3D point clouds is an alternative line of extended research.

In this section, the completed work includes Kinect's system characterization analysis, and pre-processing for 3D plane fitting. First, the sample target chosen is a flat plane

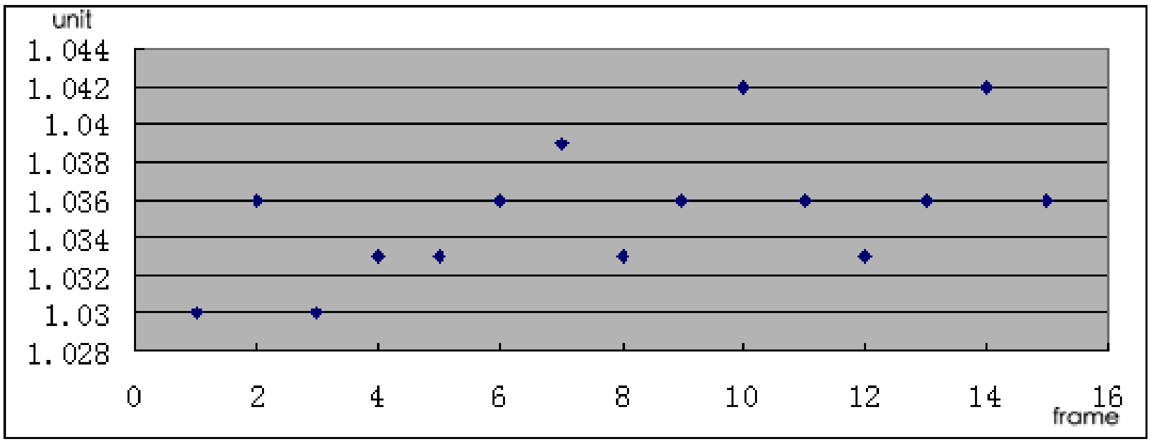

Figure 3.9: Example: depth value of pixels at the same position of the images, 15 frames, 150cm distance

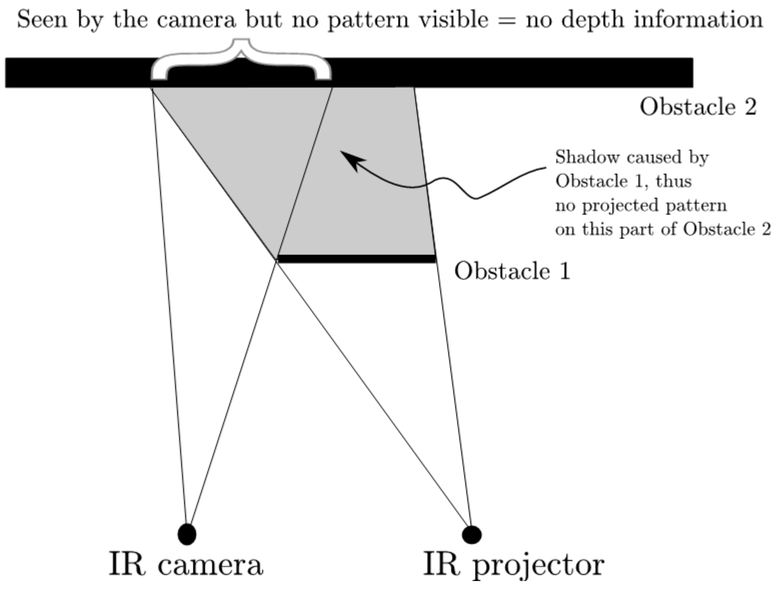
Seen by the camera but no pattern visible = no depth information
Obstacle 2
Shadow caused by Obstacle 1, thus no projected pattern on this part of Obstacle 2
Obstacle 1
IR camera
IR projector

Figure 3.10: The obstacle and multi-view fields disparity lead to the shadow effect

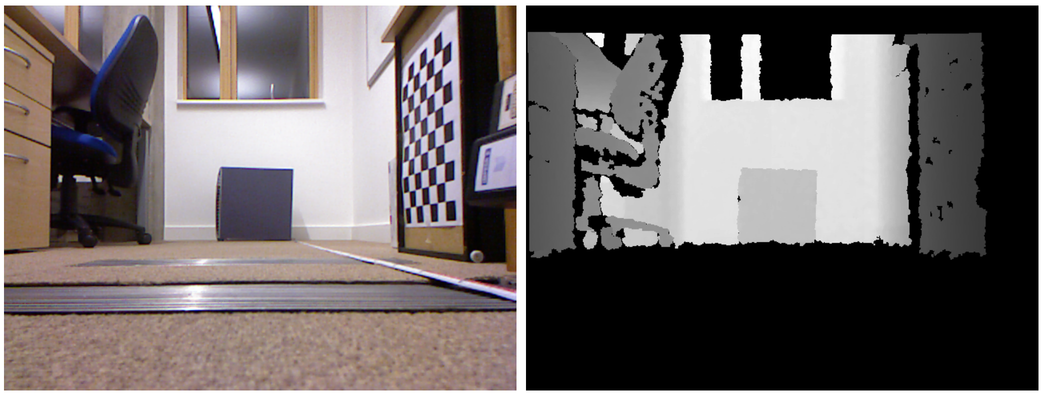

Figure 3.11: Choosing flat box as target object for error analysis

surface. As presented, the board in usual office is chosen as experiment object, shown in Figure 3.12:

The series of experiments are designed to detect Kinect's performance at different distance to a planar surface of box: Capturing 100 frames data at fixed distances to the box. Because of Kinect's independent performance in different distance, there should be at least 7 series: at 50cm, 100cm, 150cm, 200cm, 250cm, 300cm, 400cm, and also try 2 special series like 40cm and 800cm.

To calculate random errors at specific position pixels, the result can be extrapolated by Matlab funtion. For instance, in a group test of 100 frames of images, which shown on Figure 3.14. That's an example of the whole experiments.

In order to make the experiments reliable, the chosen pixels anchored in images would be agglomerated in three small size rectangles of the whole images: 1. in center positions 2. left border positions 3. right border positions

To investigate the system errors during 3D point clouds generation, the rectangles part created in point clouds will be quantitatively compared with ground truth–real ones measured and drawn on images.

The next step of experiments is about the same point in center and same point at border, still take the series of 7 distance, 100 frames, only added a third point with fixed deviation for further study. Seen as Figure 3.13:

During the experiments, some characterization of Kinect which have been observed:

(1). About pixels, such as 'Flying pixels', especially at object boundaries. The inner and outside cannot be easily classified. This is also called common distance transform (CDT).

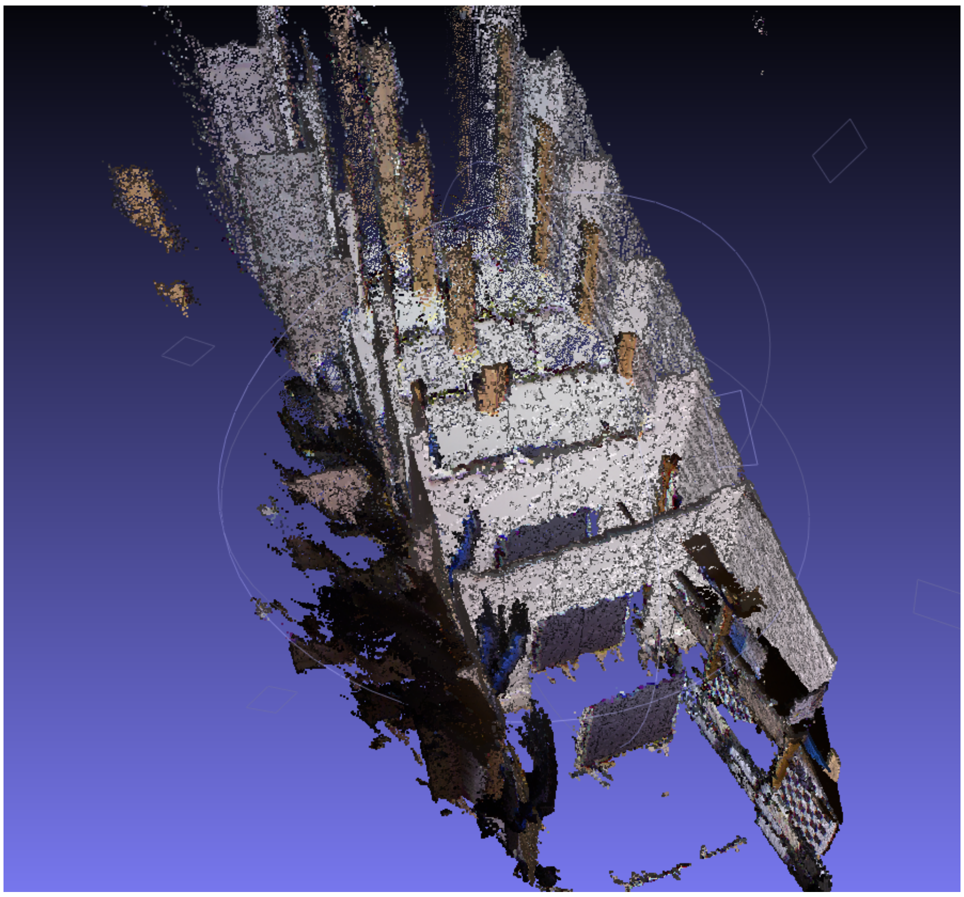

Figure 3.12: Series of experiments of Kinect's performance at different distance to a flat plane, seven groups of point clouds in an overlapping phase

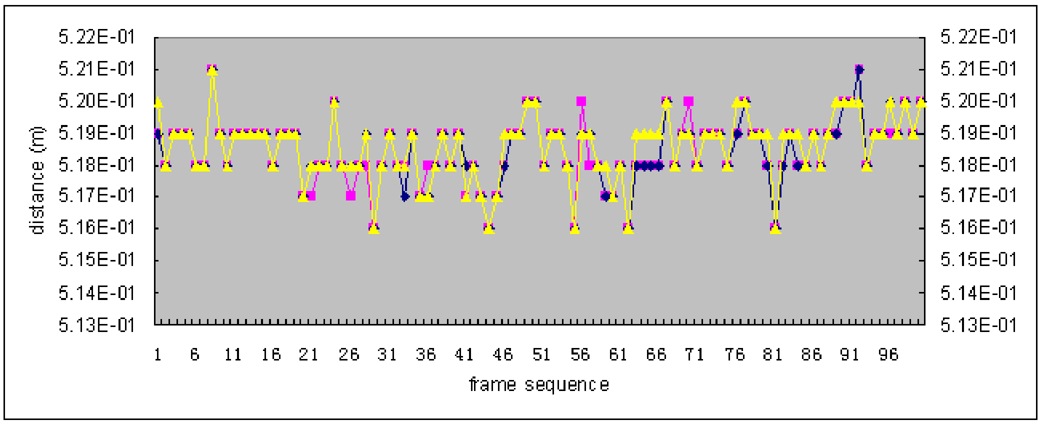

Figure 3.13: Example: depth value of pixels at same position of 100 frames images, fixed distance: 0.5m, yellow-left, purple-center, blue-right

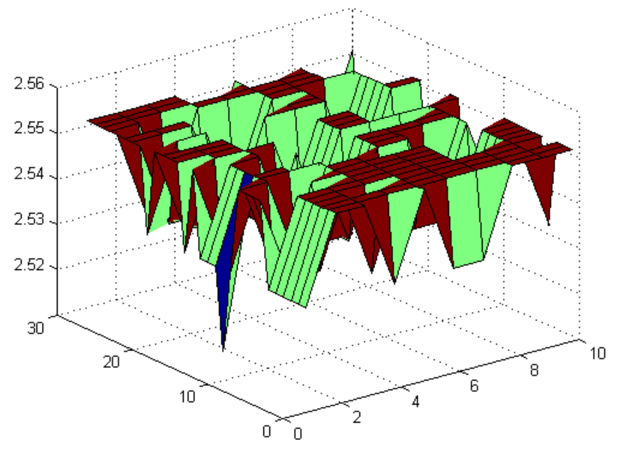

Figure 3.14: Specified of small size rectangles at one frame image, fixed distance: 2.5m

(2). Low pixel resolution at nearer distance (0-50cm), and with increasing distance the resolution gets better, at about 2 meter to the best. Then the resolution gets worse with increasing distance, until can not be recognized (more than 500cm).

(3). Individual pixels get different depth measurements, however they just "jump" (transform) between very small deviation–only two values which created by Kinect's working methods.

(4). The border data is more feasible than center data, because much more noise at the border while vision signal imputing.

A result can be derived for Kinect that always see the different resolution at different distance, and design algorithms based on these error analysis. Beyonds, the jump between CDT feature of Kinect could help to modify some detailed rules while 3D feature detection. Citing point clouds at object's center to detect distance feedback with better accuracy, while border data requires to be reconstructed for other application. The significance of "Calibration and Characterization of Kinect" is obvious. Based on the controllable error in different range, it is able to design reliable feature extraction algorithms with proper threshold value.

Modelling errors is already discussed in Section 3.1. Here are the comparisons between applied method calibration result and factory method calibration result (same center point, different methods), shown in Figure3.16 and 3.17:

Obviously presented from the quantitative results,the project's method calibration runs better than factory method dose evaluated by systematic error.

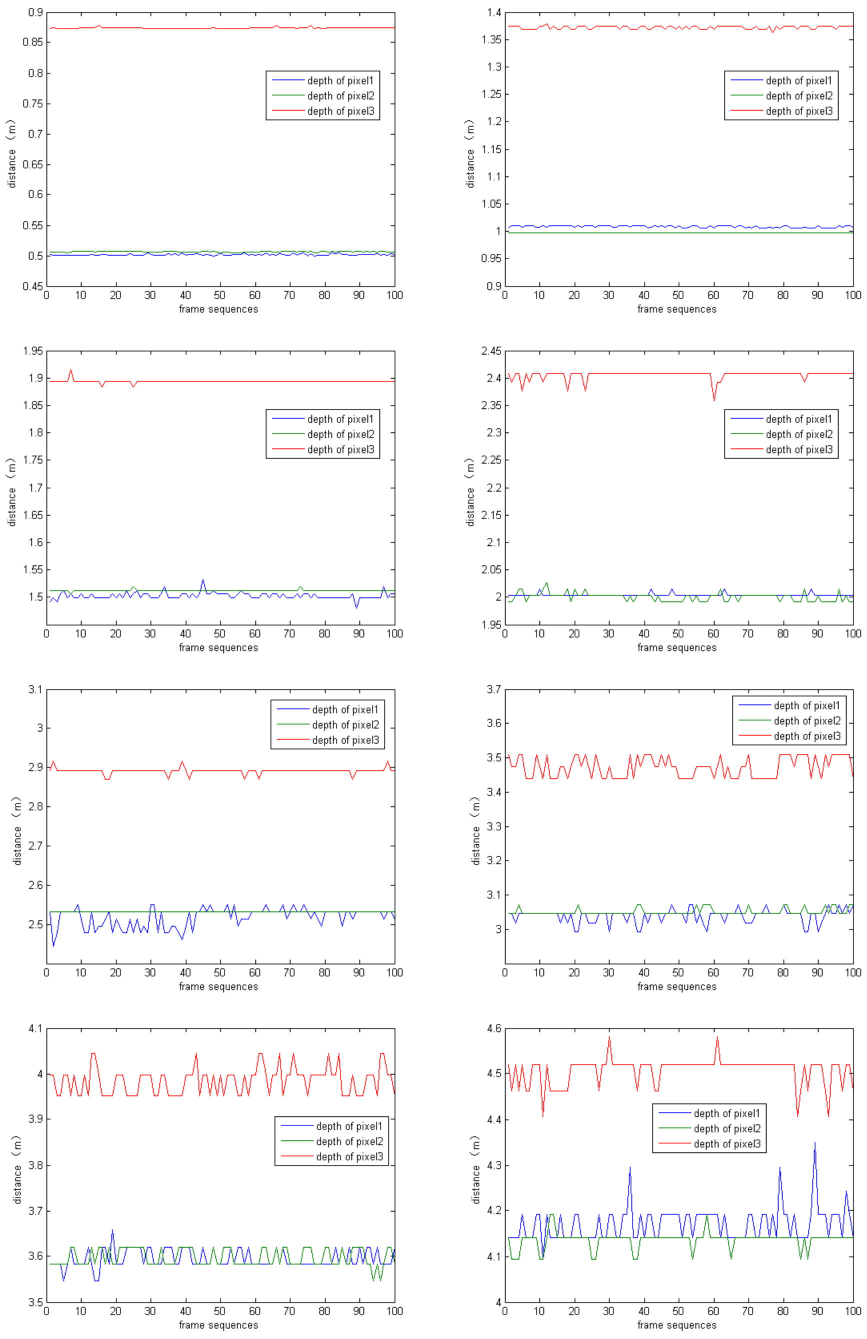

Figure 3.15: Detection of same point on plane at different distance, from 50cm to 400cm; green line-center point, blue line-border point, red line-with added deviation

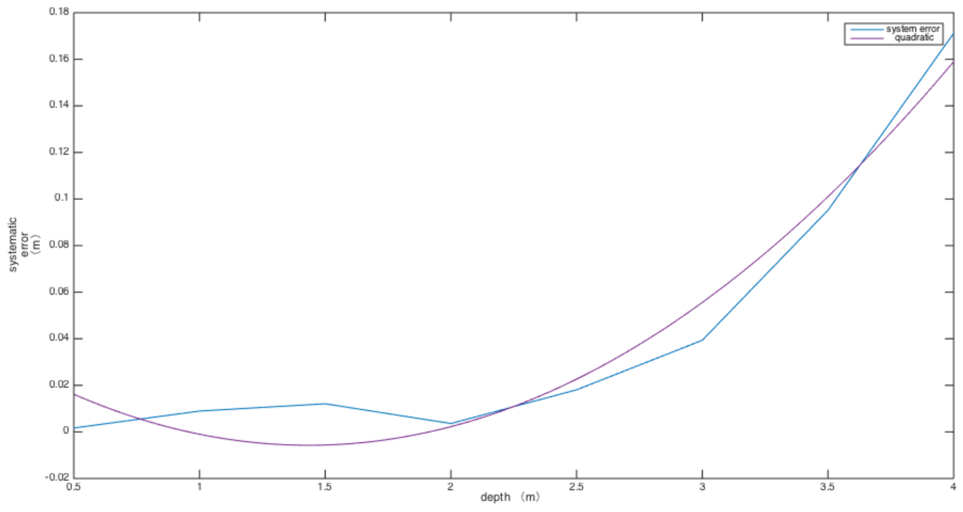

Figure 3.16: Statistics of this project's method calibration results: Center point systematic error with distances range from 0.5m to 4m

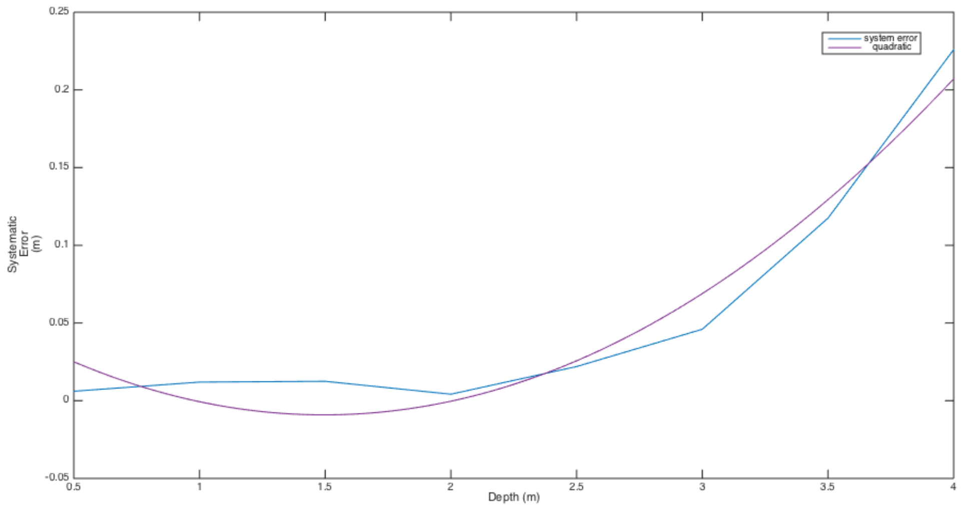

Figure 3.17: Statistics of factory method calibration result: Center point systematic error with distances range from 0.5m to 4m

Table 3.3: Standard Deviation of three pixels at the same position of the BOX, at different distance observed by Kinect

truth-value-box (cm)	border pixel (cm)	standard deviation	center pixel (cm)	standard deviation
50	50.168	0.00010576	50.64	0.000042002
100	100.9	0.00048475	99.7	0
150	151.21	0.000013336	150.25	0.003683
200	200.04	0.0055311	200.36	0.00068401
250	251.81	0.05926	253.2	0
300	303.93	0.040528	305.06	0.010286
350	359.52	0.042676	359.78	0.03992
400	417.12	0.13565	413.71	0.036342

When applied the least square fit and get the quadratic function, it can be compared with other methods discussed before. The results shows in Figue 3.18: Comparing with ideal depth measurement, both methods have bias. From 0 to 2 meters, both methods share similar performance. However, with increasing of the view distances, this project's calibration method performs less variance and bias than factory method does. This project's calibration method would lead to better performance facing large indoor room robot navigation.

The calibration results are reliable. The improved calibration method applied is indispensable, if Kinect sensor operates in navigation applications.

Another group of comparison shown in Figure 3.16 and 3.19. They are both this project's method calibration results. The difference is Figure 3.16 shows center point and Figure 3.19 shows border point.

All of the experiment data can be drawn in a scatter map shown in Figure 3.20:

3.2.3 Denoising and Filters

When simple 3D features such as planar surfaces process with fitting and segmentation with each other in point clouds, the modelled depth noises are quadratic with respect to depth, also tested by Nguyen et al [35]. The original point depth value distribution with such noises affects the RANSAC-based methods while counting of outliers. Since it is difficult to consider the noise factors into the planar surface point distribution. To cope

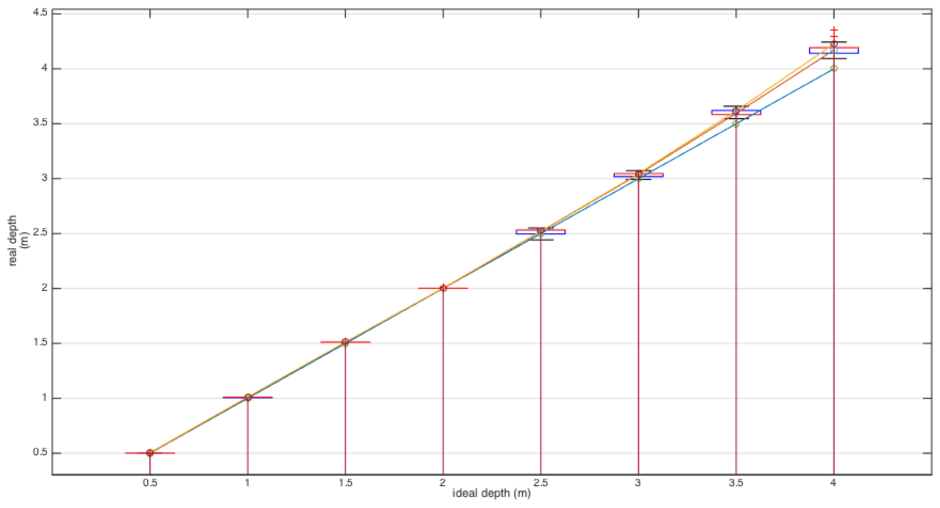

Figure 3.18: bias and variance comparison among ground truth and two calibration methods: this project's implementation and factory method

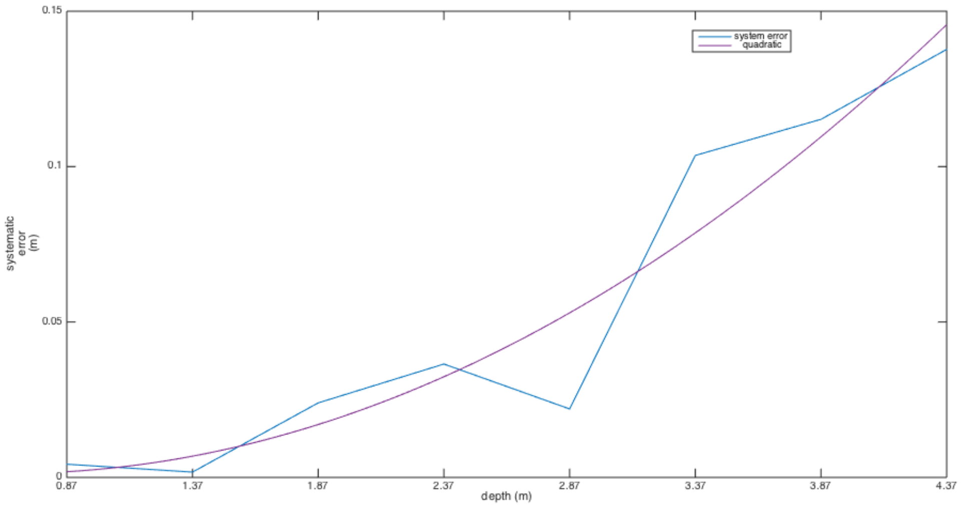

Figure 3.19: Statistics of this project's method calibration results: Border point systematic error with distances range from 0.87m to 4.37m

Table 3.4: Standard Deviation of three pixels at the same position of the WALL, at different distance observed by Kinect

truth-value-wall (cm)	pixel 3 added (cm)	standard deviation
86.75	87.425	0.00013875
136.75	137.17	0.00095814
186.75	189.4	0.00068299
236.75	240.65	0.006871
286.75	289.2	0.0056678
336.75	347.39	0.088004
386.75	398.52	0.090516
436.75	450.77	0.093182

with this scenario, it leads to another solution just to reduce the noises (pre-processing). Thus, before plane fitting, another step of filter smoothing is essential. Methods are these three: normal median filter, bilateral filter and moving least squares fitting.

By using the Kinect performance curve that correlates depth and noises, normal filtering can not simply average away noise values corrupted center and nearby pixels. It is due to the 'edges' coming from depth, which can be seen as self-blurred by linear low-pass filtering. In this case, Bilateral filtering seems to achieve a simple, non-iterative scheme for edge-preserving smoothing [38]. It is widely applied in computer vision and computer graphics.

From the sources of RGB image and depth image, the bilateral filter is designed as following:

$$I_p' = \frac{\Sigma_{r \in N} f(p-r) * g(Ip - Ir) * Ir}{\Sigma_{p \in N} f(p-r) * g(Ip - Ir)} \tag{3.19}$$

In Equation 3.19, p is the current point, r is a point in the kernel N around p, and I_p is the intensity of the point. Functions f and g respectively measure Gaussian-weighted geometric distances, and RGB color similarity. Intuitively as design, the bilateral filter tends to smooth more when neighbouring points are similar, and smooth less when there are value jumps.

Even when considering the characterization of Kinect, missing information and error segments of the scene still exist. By applying bilateral filter, the rendering result of

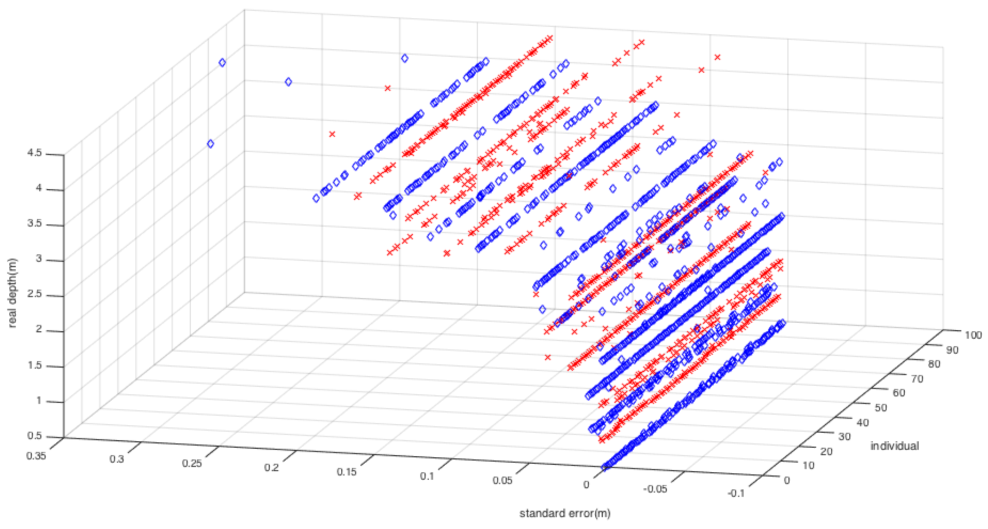

Figure 3.20: Scatter map of experiment data streams

every scene seems better. However, it rewrites the raw value output from Kinect 's data collection. As a result, de-noising using moving least squares fitting methods are proper than bilateral filter at the situation of key point collection, which the method is discussed in Chapter 4. The project pre-processing method is apt to choosing moving least squares fitting.

3.3 Summary

To build the system, the first concern is understanding the basic structure of Kinect sensors and modelling the RGB-D cameras. As a new type of RGB-Depth cameras, Kinect has its own characterization. Though the basic parameters given by the Primesense company, the sensors requires to be calibrated before its implementation in SLAM. The proposed work contains the progressively objective: modeling and calibration of all the RGB camera, IR camera and IR projector. In the assumption that IR camera and IR projector combine a system of stereo cameras, the pinhole model is also adoptable. The specific factors affect the modelling validity can be added later, guided by the Primesense's prototype. Calibration of all these optical components are proved absolutely necessary. Furthermore, the relation between depth detection mechanism and speckle pattern emitted by IR projector is explained by equations during the modelling and system calibration. By using Kinect's special speckle pattern and normal calibration theory, the project makes a hypothesis that repeatable approaches of Kinect calibration can be quick, efficient and accurate enough.

To test the hypothesis, many calibration methods are introduced. The performance of IR projector presents a special case comparing with normal camera system. Thus it would be helpful to modelling and calibration by quantifying the speckle pattern disparities with the depth data given by the sensors. Designed methods combining normal calibration methods and modification for Kinect sensors are valued by repeatable experiments. If the modelling and calibration is correct, it will increase the depth data accuracy. These appraised approaches achieve reasonable point clouds comparing with automatic calibration. While automatic calibration method only scales to Kinect's entertainment application, now it is improved and can be performed in SLAM and robot navigation. The further experiments are designed as comparing the real objects and the point clouds re-projection with calibration parameters.

The chapter presented both theoretical and experimental analysis of the geometric quality of Kinect's depth data. From the results of calibration and error analysis the following main conclusions can be drawn:

- To eliminate misalignments between the colour and depth data, accurate modeling and stereo calibration of the IR camera and the RGB camera is necessary;

- The systematic error of depth measurements increases quadratically with increasing distance from the sensor;

- Noises and ramdom error increases at object's edge area, denosing methods are needed;

- Comparing original calibration method, quick and reliable calibration method in this project can improve the performance of Kinect sensor in mobile robot navigation.

Chapter 4

3D plane fitting algorithms design

This chapter starts with software outlook overview based on the study about Kinect sensor itself. Secondly, Method and algorithms for 3D plane fitting are presented, both RANSAC and Hough Transform method. The design of the algorithms and results are shown as the final part in this chapter.

4.1 Software Missions Overview

4.1.1 Understanding the nature of Kinect sensors data

Following the discussion in third Chapter's summary, the overall accuracy of the system depends on a variety of factors. During appraising of different approaches, the depth data and color data measured by the Kinect offer with more information, which can be argued for further implementation. During 3D features proposed for object recognition, it is significant to distinguish the systematic error coming from the modelling and calibration, and the random error which describes the characterization of Kinect. By Repeating the experiments the project manages to solve this problem. Except that, official claimed resolution and accuracy of the Kinect device can help to judge if the 3D object recognition (such as planes) are correct.

Experiments to detect real world objects are classified as two or three kinds. Some combine intensity images with depth for object recognition while some others focus on 2D/3D hybrid approaches. For this project, it is better to define the problem from easy geometry objects, then upgrading to complex environment step by step. The first challenge is to detect flat plane from several distances with fixed intervals. Another issue is about the curve detection and re-projection, which can pick a round cup or cylinder pillar as

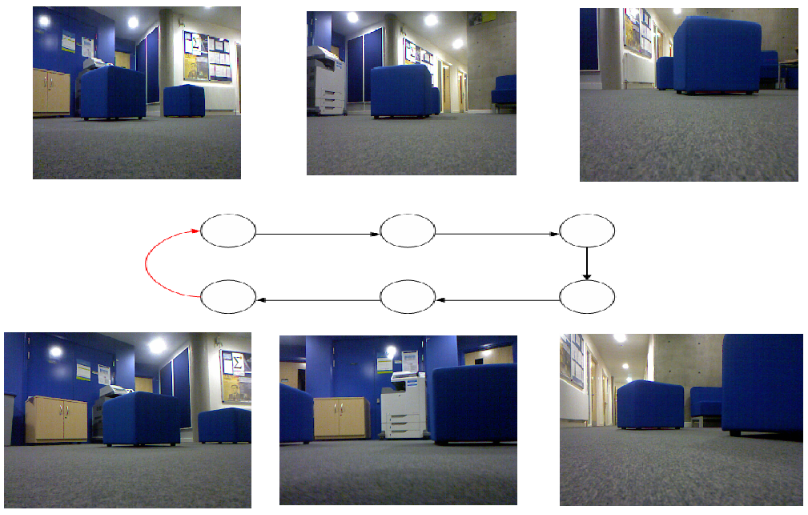

Figure 4.1: Loop closure matching frames, real pictures and graphs

target object.

4.1.2 Designing feature extraction and registration algorithms

After the building of 3D point clouds and its accuracy for SLAM application justified, the next step is to apply the original data source and design algorithms operating for robot navigation. The aims are set to define the mapping content, build the maps in avoid of problems such as Loop Closure problem which shown in Figure 4.1, and evaluate the quality of maps. There are three key requirements during we consider the target of SLAM problems and plane fitting challenge.

1. The feature extraction has to be real-time and robust. It requires that the ICP registration algorithm operates in a high speed.

2. Develop better algorithms to solve the Loop Closure problem. This target is primary in SLAM solution and there are many proposal plans. The point clouds with position information and color images with texture features combines with proper weights is important. Besides, no easy and universal methods exist to meet varying, novel requirements. A specific method to apply the depth data source of Kinect for this project's platform is the key point. With the algorithm study, some rules to extract the key interesting points in point clouds are established. 3D point clouds feature detection, description and matching is a new developed field, especially for applications like Kinect using in SLAM. Applying the 2D image features algorithms for reference would provide some novel ideas

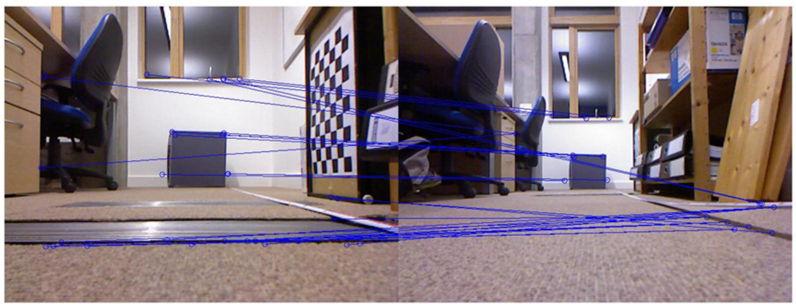

Figure 4.2: Before Kinect's 3D point clouds scenes, 2D SURF without threshold control takes samples of interest points

to 3D research area.

3. To cope with the large sized data coming from the sensors, preprocessing is essential. The organization of software and hardware source brings out the efficiency problem of using point clouds. It affects the results SLAM solution to an extent.

As reviewed in the Chapter 2, new feature detectors and descriptors are normally created as varieties of classic approaches. The domain of our project scopes mainly to the 3D feature extraction methods. The aim is to evaluate if novel designed detection and description methods can deal with Kinect's 3D point clouds scenes. The improvement is focused on both speed and accuracy. By using spin images and concepts from SIFT, new designed descriptors need to satisfy the standard that feature invariant in scale and in rotation. To test the algorithms' efficiency, it can be put in to a benchmark and compete with exist algorithms. There are lots of benchmark data for the algorithm efficiency testifying. For the distinctive data of Kinect sensors, benchmarks are need to be constructed by large amount of scanning work.

For instance, Speeded Up Robust Features descriptor(SURF), which is similar to SIFT, increasing robustness and decrease computation time. research in 2D SIFT and SURF are maturely developed. By applying SURF algorithm first to take samples of the interest points, which will be benefit for accurate interest point registration and fitting in 3D point clouds. With proper control in threshold, the picked out points of interests are more reliable, shown in Figure 4.2 and Figure 4.3.

Except of the efficiency debate, the algorithms still have to run through a series of tests.

1. How do the ICP algorithms operate when it faces the Kinect sensors' noises. Light

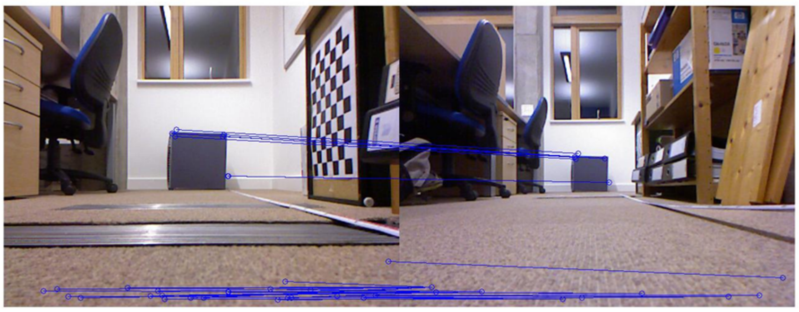

Figure 4.3: Before Kinect's 3D point clouds scenes, 2D SURF with threshold control takes samples of interest points

reflection phenomena and luminous source placed in the scene are one kind of noises, while another kind of noises come from movable objects like walking people during mapping.

2. How to deal with the content of maps? In indoor environment scenarios, data can be recognised and classified to save more spaces. It needs the background knowledge, such as ground surface recognition. Data formatting is important considering the computability. Data formatting simplification may speed up the whole procedure if chosen properly. It requires different types of compressed data formatting evaluation.

3. The implementation of a simple mobile robot system with Kinect sensors. Creating a prototype, examine the whole project with real environment ground truth is significant for evaluating the quality of maps building.

4.1.3 Introducing multi-objective optimal searching methods to registration

For the designed feature extraction methods and matching algorithms in Kinect's 3D point clouds scenes, optimal association and management of these approaches would provide better results. It is common to use more than one feature, for example to apply 2D image feature and 3D point clouds feature together for registration, which presented in Figure 4.2 and Figure 4.3. Other strategies include combining shape-valued feature and color-valued feature together, real-time feature and feature with prior knowledge restriction together, etc.

This project makes a hypothesis that registration combining the proper features together would improve the quality and accuracy of the 3D point clouds by using the optimization of a multi-objective method. To determine whether it successes or not, experiments are designed as four sets:

1. Registration with one single feature, like only color; In experiment it applies poster on wall;

2. Registration with another single feature, like only depth; In experiment it applies white bookshelf;

3. Registration with both features in separate methods; In experiments it applies the whole side of room;

4. Registration with both features in an optimized method considering both objectives' searching; In experiments it applies the same side of room.

Poster is rich in color feature, however poor in depth feature. Bookshelf is rich in depth feature, however rich in color feature. Approaches only one feature or separate methods can not cope with complex environment with all kinds of objects. The experiment also should evaluate the result from many other aspects: results accuracy, robust with different environment, processing time consumes and source consuming.

Considering ICP algorithms, it relates with searching and optimization by its nature. A hypothesis that introducing non-linear optimization method into ICP would improve its performance in 3D point clouds matching is also listed as the objective in this stage.

Multiple view geometry in computer vision and standard calibration are supporting the studies of camera modelling and calibration. For registration part, the class of ICP algorithms are the most important techniques to be used. When dealing with feature matching, referenced ORB, SIFT, SURF, Spin images methods provide novel ideas. Further, combination of algorithms and multi-target optimization would make the designing more flexible and robust.

Multi-objective optimization(i.e., Newton-Raphson method, stochastic searching optimization, non-linear programming) are widely applied in algorithms similar to Convolutional Neural Networks (CNNs). Feature engineering is apt to understanding 3D point clouds as multi-objective clusters, while SLAM and geometry methods are apt to accurate matrices and transform computation. To summarise the trend, supervised learning helps SLAM systems geometrically understand the environment immediate and build associations across distinct object instances.

With the introduction and simple experiment of the project's hypothesis, challenging ideas emerge themselves and are not that hard to understand in simple scenarios. In the same way, the project choose a simple planar surface object for example during demonstration on algorithms.

4.2 Planar Surface Fitting Methods

For Kinect's computational geometry and feature detection in maps, planar surface recognition and fitting is an essential and basic task to deal with 3D point clouds. With the data provided by Kinect sensor, clouds of 3D points are scattered in the space. During plane construction process, good estimation is significant to give a planar form $(x, y, f(x, y))$. Its estimated normals which represent the surface form, can be valued as the key point, deciding whether the estimation is good or not. "In surface reconstruction, the quality of the approximation of the output surface depends on how well the estimated normals approximate the true normals of the sampled surface" [10]. The fitting of plane is kind of a local normal (planar form) estimation, then proper scales of point clouds segmentation and classification.

In real world, 2D fitting is about linear form, while in 3D is about plane fitting. The study of 3D plane fitting can be tracked back to the year around 1990. Researchers started to notice the scattered data on a special domain, which the plain model is just the planar surface domain. Hoppe et al. of University of Washington developed an algorithm for 3D surface construction with or without boundary, using a set of unorganized points on or near the surface. The paper not only gave the popular methods for fitting, also termed closed surface, bordered surface and simplicity surface [19]. The famous methods for plane fitting includes fundamental Least Squares (LS) method, Principal Component Analysis (PCA) and RANdom Sample And Consensus (RANSAC) algorithm.

As the important steps of estimation point segmentation, there are two problems which need to be considered while methods designing: how to define outliers and how to reduce sensitivity of outliers comparing with points in the plane form. The Least Squares (LS) method is simple to understand: with every (x_i, y_i, z_i) in a sample set, the characters of A,B, and C can determine the plane $z = Ax + By + c$, while the sum of the squared errors of $'Ax_i + By_i + C - z_i'$ is minimized. To solve to gradient linear equations, the characters are easily found.

It is easy to find the solution of LS method too sensitive to outliers and make itself not reliable. Followed that the Principal Component Analysis (PCA) faces the similar problem. PCA is a statistical technique that is typically used to identify a small set of mutually orthogonal variables which explain most of the underlying covariance structure of

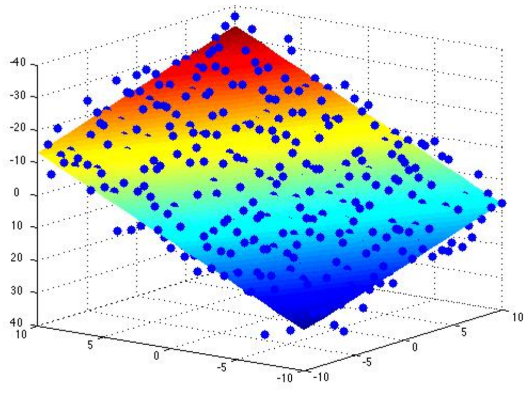

Figure 4.4: Least Squares method for 3D plane fitting

a dataset. [36] As an old powerful tool in exploratory data analysis and making predictive models, the 3D-PCA version is just proper for planar surface construction. However, due to the affection from observation, it is still not robust. Many kinds of PCA related approaches are developed, and further optimization is kept on.

RANSAC is an abbreviation for "RANdom SAmple Consensus". The algorithm was first published by Fischler and Bolles. As an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers, the probability for reasonable result increases with more iterations are proceeded. Because of the iteration, "RANSAC is very efficient in detecting large planes in noisy point clouds but very slow to detect small planes in large point clouds". [9]. Usually for Kinect's application on SLAM, the situation is that planar surfaces take outright majority of the compositions. Large noisy planes like ground and wall are easy to detect using RANSAC. However, when facing small open style objects, the weakness of computation complexity is kinds of fatal.

For other methods, there are still some ideas, such as region growing and Hough Transform algorithms. Hough transform method is wildly used in 2D for shapes (line, circle) detection. While target is to focus on the detection of planes in 3D point clouds.

Hough transform can be used for the detection of 3D objects in point clouds. The extension of classical Hough transform for plane detection is quite straightforward. Surface detection using Hough Transform in 3D just likes line detection in 2D, simply adds another variance to the Equation 4.1.

$$\rho = xcos[\theta]sin[\phi] + ysin[\theta]sin[\phi] + zcos[\phi]; \theta \in [0, 360), \phi \in [-90, 90] \qquad (4.1)$$

For example, in N-point point-cloud, every point is voted in $m[\theta] * n[\phi]$ bins. To run 3D Hough Transforms, it needs total voting $N * m[\theta] * n[\phi]$ times. Thus computing complexity of standard 3D Hough Transform is huge. Algorithm runs slow with less efficiency. The designed histogram algorithm does not calculate the complete Hough Transform for all points. The randomized selection three points method leads to a proper Randomized Hough Transform (RHT) method [5]. The designed method cleans data first, pre-processes data by getting lower resolution of the point clouds. Then it randomly selects groups of three non-collinear points. Three points counts one single vote to the plane detection cells. This strategy significantly reduces the voting cost. Except that, for multi-plane situation, such as different depths, RHT can complete the estimation in one iteration and save storage for the planes.

For all these methods, learning to transferring them to cope with Kinect sensor implementation is important. In Kinect 3D point clouds scenarios, border and edge data, even though blur, still can be calculated with multi-plane clustering. Setting threshold of peak voting detection will recognise large planes like ground and walls fast. Applying with knowledge background of objects and environment, algorithms can be improved in speed and robustness.

When considering with the standard organization of this algorithm, it always turns the target-problem-framework into three steps, which repeated in an iterative. For planar surface fitting, RANSAC is applied to identify each points in the sensor dataset, recognize that if a point belongs to a plane and give the parameters estimation.

1. The first step is to describe the plane model function and find the minimal sample sets. Three points not in a line can define a plane. Data collection starts with randomly selecting three points from the dataset.

2. The second step is to compute the plane model parameters, by the selected points coordinates in a minimal sample set. Actually, this minimal sampling method is where RANSAC is different to other approaches. For example Least Squares method, the estimation of plane function parameters have to use all the data available.

3. In the third step, it is needed to check the consensus set of a plane, which contains as more proper points as it defines. During the iterations, a final consensus set is resulted and compared to a new set. The optimization simply count the inliers amounts and return to a better performed consensus set. The iteration threshold also depends on the dataset which is found to contain most inliers elements.

RANSAC can be sensitive to the choice of the correct noise threshold, which makes the data distribution important while selection step. It would be highly valued if the algorithms is designed for special application or special sensors, as an example work by Sanchez and Zakhor [44]. Looking for a optimized solution for Kinect sensor and SLAM application, is the goal of this project.

4.3 Improved RANSAC Algorithms and Experiments

As mentioned before in introduction, RANSAC and Hough Transforms are two of the most popular methods for planes in point clouds construction and identification. With development of 3D point clouds, many applications developed in the new research area in last decades [56], [57], [36].

The algorithm design of RANSAC for 3D planar surface fitting is implemented with help from the Point Cloud Library (PCL) [43]. In PCL, many open source point cloud processing modules such as **pcl-segmentation** and **pcl-sample-consensus** make the fitting and segmentation easier. When given sampled representation from Kinect, the whole frame or part of frame is observed as an unorganized point cloud. Since the dealing with whole clusters may become too slow, this project only make a part of the planar board as input. After the RANSAC algorithm, the largest amount of inliers datasets would be all considered for the equation of this target plane. An added changeable deviation threshold can control the speed if there's high accuracy requirement.

That's the first step of the experiment. In Table 4.1, a dataset of 1784 points of one board, several frames results are showed (dataset also showed in Figure 4.5).

Evaluation of this experiment is simply managed by Confusion Matrix, a special kind of contingency table, with two dimensions. Since the target plane is perfect designed as plane ground truth(at least in accuracy of mm level), the table would be easier to draw. All of the 1784 points consturcted the true data, means there is no false data in this experiment. Here iinlier points are the real positive cases in the data (TP), while outliner points are the real negative cases in the data (TN). false positive cases (FP) and false negative cases (FN) equal to 0. The percent of TP points in 1784 points is defined as accuracy (ACC).

If sensor accuracy accepted in coarse level (the threshold is set as 5cm), fixed board with distance around 1.8 meters would always feedback the same result. The only difference is amount of the inliers. It is due to a few of points jumping affected by speckle pattern across different frames.

However if setting the sensor accuracy upgrading to more accuracy fitting, such as 1cm threshold, RANSAC output datasets cover the entire target surfaces shows not such consistent across frames. Because of the edge shadow effect, and the non-depth blanks segmentation due to single frame detection, the same plane in real world may result with little different.

From this experiment, the top acceptable value of accuracy (ACC) is around 98.5%. Further experiments coping with planes and other objects, there will be false positive (FP) and false negative (FN) data in the sets. For example, to detect a desktop surface with several books and cup on board. Then ACC and other evaluation index of Confusion Matrix will present more aspects about RANSAC algorithms operation.

Compared with percentage result, it displays that if threshold is too large, as 5cm. Then all the hypotheses sampled dataset tend to be ranked equally. On the other hand, when the noise threshold is too small, the estimated parameters tend to become unstable. RANSAC method's sensitive to the choice of the correct noise threshold [55], and better adjusted threshold would show a balance in both accuracy and speed for this application.

Even though setting the same threshold, RANSAC runs on different frames data results different amount of inlier points. It depends on border points with unreliable values. Sometimes border points even feed back with no depth value (shown as black in Figure 4.5 and 0mm in depth). It is believed that these flexible points make the RANSAC results different with same threshold.

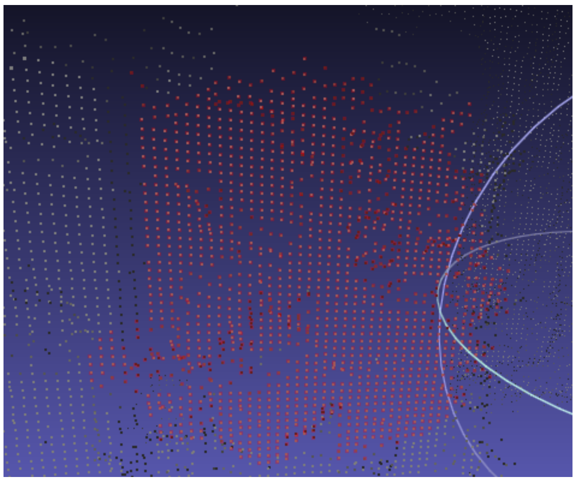

Figure 4.5: Dataset of 1784 points of one planar board, pick up in a 3D software screen

For frames containing more than one planes, RANSAC can still get the right segmenta-

Table 4.1: Statistics of different frames about RANSAC processed dataset

Frame Number	Threshold of RANSAC	Amount of Inlier Points	Percent
Frame 1	1cm	1706	95.6%
Frame 2	1cm	1687	94.6%
Frame 3	1cm	1712	96.0%
Frame 4	1cm	1660	93.0%
Frame 5	1cm	1677	93.9%
Frame 6	5cm	1757	98.5%
Frame 7	5cm	1762	98.8%
Frame 8	5cm	1748	98.0%
Frame 9	5cm	1747	98.0%
Frame 10	5cm	1755	98.3%

tion for each, but with the amount of inlier points reduced below 90%. Datasets at border regions are always contained by noises, which need pre-progress filtering. In a coarse level, the algorithms could compute the overlap part between two planes and do simple dataset operation. Meanwhile, the basic procedure could be improved from several aspects. For example, the estimated area of planes decides the selection of points. If choosing the points as a previous set rule, it will avoid large amounts of useless groups, for example in a line, too compact or too scattered. With the sections divided, the group of three points in a line can be perfectly avoid. To check if the plane is a local one, make sure the group of points will never be selected again, there can be an iteration inside the divided section.

Hough transforms are a well-known cluster algorithm of image transforms with history, which can identify parametrized objects of a certain class. For Hough transform, it is more like a histogram principle, which can put every possible plane features into the bins, and calculate the majority. In this method, the planes of different range can be found together, if the threshold is set properly. However, during the experiments, even the ideal plane may arrive to a doubtable answer because of the sensitivity to the threshold. Here we assume the phenomena come from the affection of Kinect's characterization itself, for intermittent value distribution.

To combine those algorithms together, it requires some support processes. The designed structure is about detection of planes with Randomized Hough Transform, and use

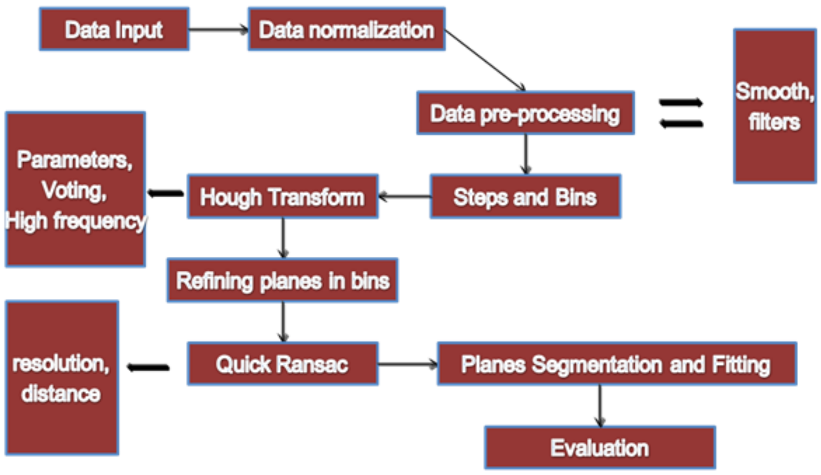

Figure 4.6: Coarse and fine resolution fitting algorithm combined with both RHT and RANSAC

RANSAC with shape background knowledge to do the iteration and give the estimation of planes, finally by Least Squares method to obtain an plane with better accuracy. Considering the Kinect's characterization–resolution change with distance, the plane detection may lead to a faster way if getting the histogram also changeable with resolution. Make the Hough Space suit to two or three level of resolution, may properly work for Kinect sensor data. To implicate the multi-resolution plane, other methods such as interpolation, Bayesian approaches, and filter methods may help to achieve this goal.

The coarse and fine resolution fitting algorithm combining both RHT and RANSAC is the target. The concept of coarse-fine strategy has several advantages over a single resolution segmentation [37]. For Hough Transform, the advantage of detecting multi-planes would improve the efficiency for whole frame segmentation. Large planes such as ground and walls are quickly separated. Then the RANSAC for high resolution and high accuracy algorithm would run on the bases of exist planes and precise compute the function expression of each plane. With this method, which only sensitive to planar surfaces in a point cloud dataset, an elemental feature detection method for Kinect makes the most common features in real world segmented. For complex features in high level, non-planar segments need some other feature extraction methods.

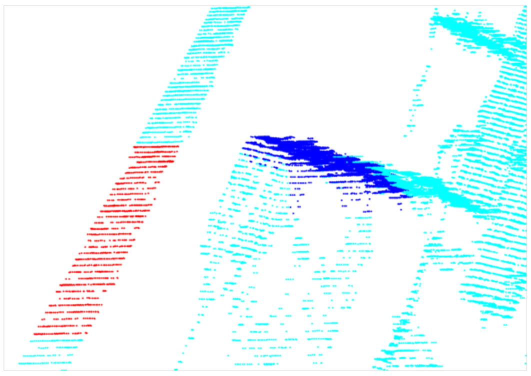

Figure 4.7: Multi-planes extracted from office 3D point clouds

4.4 Summary

In this chapter, this project mainly discuss about 3D plane fitting problem. Based on the controllable error in different range, we can design reasonable feature extraction algorithms with proper threshold value. The input stream data as raw RGB-images, go through combined algorithms, which include pose estimation by RANSAC and pose graph improving by Hough Transform, output as coloured 3D point cloud models.

There are 5 conclusions retrieved from the results:

1. From the points cloud data collected in office environment, most objects are shown as plane surfaces, which can be segmented by planes detection algorithms.

2. Hough Transforms and RANSAC can be implemented to detect multi-planes in point clouds separately. Improved algorithms are able to solve even more complicated figures, such as bookshelf environment.

3. However, during the experiments, the ideal plane such as whiteboard may lead to an unreliable answer because of the sensitivity to the threshold. Here it is assumed the phenomena come from the affection of Kinects characterization itself, for sparse value distribution. The analysis is missing.

4. The designed structure contains detection of planes with Randomized Hough Transform, and RANSAC with shape background knowledge to run the iteration and gain the estimation of planes, finally applying Least Squares method to obtain an plane with better accuracy. The structured algorithms are successfully implemented in three parts separately. Combined algorithms test and results evaluation are needed.

5. It is a novel method of combination of two algorithms together for plane detection. After data cleaning, fast RHT with peak thresholds works well for multi-planes. Many

installed parameters are not generalized perfect for all kinds of environments like different scales of planes together. System robustness evaluation part should be added, such as example explanation, and algorithms results comparison with ground truth.

Chapter 5

Conclusion and Future Work

This chapter starts with conclusions of the whole thesis. It lists what has been done and some reflection on understanding of RGB-Depth cameras. Secondly, contributions and achieved results are shown, both from modelling and algorithms. The final part talks about future work and trend.

5.1 Conclusion

The thesis aims to prove the hypothesis that Kinect can be applied in SLAM field in an indoor environment.

After overview the SLAM theories and their development, this thesis raise a project to build a system of sensors (hardware) and algorithms (software) together implementation for SLAM.

1. The project's key issue is Kinect: Study of its characterization and modelling the cameras sensor. The study makes it an clear answer that: After modelling and calibration, Kinect can be used in navigation as visual cameras systems, with plus range information.

Although this project's model shares the same idea of pinhole models, this design from modelling to calibration has differences and advantages. By analysing the distribution of systematic error, it shows a clear procedure of calibration and evaluation method for Kinect and other similar commodity depth cameras. It proves that depth cameras like Kinect, the noise level is a quadratic function of the depth. It also discusses solution about raw data preprocessing and de-noising, improving the quality of rebuilt environment depth maps.

What the project pursues is a better modelling of the general depth transform mechanism, and a better understanding of the depth cameras' noise model. It results a proper accurate model for kinect depth value, also a quick application method of depth cameras devices.

2. Designing and combination of algorithms can in a degree improve Kinect performance during coarse and refined 3D registration of pairs of images. Feature matching in single plane and multi-planes are solved as cluster problem using RANSAC and Hough Transforms together. The combination shows better performance of Kinect than 2D SURF detection and recognition, and better than two separate algorithms in complex environments.

The proposed novel algorithms learns to generate threshold parameter and align them jointly. By employing RHT and RANSAC, multi-planes can be accurately segmented in large scale. The number of planes wanted in one frame can be pre-set. It also combines shape knowledge and dropout sub-sampling methods to speed up the algorithms. Though facing different environment, the whole pipeline still works if threshold and parameters set correctly.

3. Further conclusion is about understanding of building systems: which includes 6 valuable poles: modelling, structure or pipelines, data, algorithms, experiments and evaluation results.

5.2 Future Work

System building is not only pieces of work piled together, but also combination and glued in logic. The reliable and robustness of projected Kinect SLAM system needs more experiments and evaluation.

Further more, improvement of sensor-environment adaptation: in complex environments there are more curve surfaces and non-rigid surfaces to deal with. Also an empty room (for example only with white walls) needs more basic geographic features to be detected and applied for navigation.

In recent years, as Deep Learning develops, unsupervised calibration method become possible. In the future, end-to-end learning and transfer learning would recreate Robot's SLAM. An even general methods of indoor environment SLAM can be quickly solved by real-time system. As people define features, rewards and adoptions, modelling parts would efficiently build a learning system and optimize itself to the target. During 2015 and 2016, there are already some innovative papers in this area coming to the research community, and some implementations to the engineering world.

Another looking forward section is about semantic recognition. Integrating semantic information into SLAM is more and more popular in 2D and 3D. There was a lot of interest in incorporating semantics into todays top-performing SLAM systems. One day in the future, the new video SLAM-Convnets like ConvNets database benchmark would be widely used.

The next level of application is not only robots, but auto-mobiles. Actually this technology of navigation has been developed for several years. Drones are also nice platforms for RGB-depth cameras, and the piloting is a challenging device-environment interaction.

References

[1] Tim Bailey. Constrained initialisation for bearing-only slam. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1966–1971. IEEE, 2003.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[3] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

[4] Paul J Besl, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[5] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):3, 2011.

[6] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 138–145, Mar 1985.

[7] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[8] Andrew J Davison, Yolanda Gonzalez Cid, and Nobuyuki Kita. Real-time 3d slam with wide-angle vision. *IFAC Proceedings Volumes*, 37(8):868–873, 2004.

[9] Jean-Emmanuel Deschaud and François Goulette. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing. In *3DPVT*, 2010.

[10] Tamal K Dey, Gang Li, and Jian Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Point-based graphics, 2005. Eurographics/IEEE VGTC symposium proceedings*, pages 39–46. IEEE, 2005.

[11] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[12] H.F. Durrant-Whyte. Uncertain geometry in robotics. pages 23–31, 1988.

[13] 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*, volume 180, pages 1–15, 2011.

[14] Nicola Fioraio and Kurt Konolige. Realtime visual and point cloud slam. In *Proc. of the RGB-D workshop on advanced reasoning with depth cameras at robotics: Science and Systems Conf.(RSS)*, volume 27, 2011.

[15] A Flint, A Dick, and A Van den Hengel. Local 3d structure recognition in range images. *IET Computer Vision*, 2(4):208–217, 2008.

[16] Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y Ng, and Daphne Koller. Integrating visual and range data for robotic object detection. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*, 2008.

[17] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[18] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. *RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[19] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[20] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[21] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[22] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. *Computer vision–ECCV 2010*, pages 589–602, 2010.

[23] Kurt Konolige and Patrick Mihelich. Technical description of kinect calibration. *Tech. Rep., Willow Garage*, 2011.

[24] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Labeling 3d scenes for personal assistant robots. *arXiv preprint arXiv:1106.5551*, 2011.

[25] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4007–4013. IEEE, 2011.

[26] Thomas Lemaire, Cyrille Berger, Il kyun Jung, and Simon Lacroix. Vision-based slam: Stereo and monocular approaches. page 2007.

[27] Thomas Lemaire, Simon Lacroix, and Joan Sola. A practical 3d bearing-only slam algorithm. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2449–2454. IEEE, 2005.

[28] John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447. Ieee, 1991.

[29] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[30] Ajmal S Mian, Mohammed Bennamoun, and Robyn A Owens. A novel representation and feature matching algorithm for automatic pairwise registration of range images. *International Journal of Computer Vision*, 66(1):19–40, 2006.

[31] Benjamin H Morse and Howie Choset. Adaptive data confidence using cyclical gaits on a modular snake robot. 2011.

[32] Philippe Moutarlier and Raja Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I*, pages 327–346. Springer, 1990.

[33] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[34] Paul Newman, David Cole, and Kin Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187. IEEE, 2006.

[35] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012.

[36] Abdul Nurunnabi, David Belton, and Geoff West. Diagnostic-robust statistical analysis for local surface fitting in 3d point cloud data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science, Volumes I-3*, pages 269–274, 2012.

[37] Bastian Oehler, Joerg Stueckler, Jochen Welle, Dirk Schulz, and Sven Behnke. Efficient multi-resolution plane segmentation of 3d point cloud. pages 145–156, 2011.

[38] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *Computer Vision–ECCV 2006*, pages 568–580, 2006.

[39] Silvia Rodríguez-Jiménez, Nicolas Burrus, and Mohamed Abderrahim. 3d object reconstruction with a single rgb-depth image. In *VISAPP (2)*, pages 155–163, 2013.

[40] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[41] Michael Ruhnke, Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Range sensor based model construction by sparse surface adjustment. In *Advanced Robotics and its Social Impacts (ARSO), 2011 IEEE Workshop on*, pages 46–49. IEEE, 2011.

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

[43] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[44] Victor Sanchez and Avideh Zakhor. Planar 3d modeling of building interiors from point cloud data. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1777–1780. IEEE, 2012.

[45] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[46] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[47] Sebastian A Scherer, Daniel Dube, and Andreas Zell. Using depth in visual simultaneous localisation and mapping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5216–5221. IEEE, 2012.

[48] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2051–2058. IEEE, 2001.

[49] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735–758, 2002.

[50] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Consumer depth cameras for computer vision*, pages 3–25. Springer, 2013.

[51] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[52] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[53] Marcus Svedman, Luis Goncalves, Niklas Karlsson, Mario Munich, and Paolo Pirjanian. Structure from stereo vision using unsynchronized cameras for simultaneous localization and mapping. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3069–3074. IEEE, 2005.

[54] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4):253–271, 1998.

[55] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[56] Roland Wahl, Michael Guthe, and Reinhard Klein. Identifying planes in point-clouds for efficient hybrid rendering. 2005.

[57] Jan W Weingarten, Gabriel Gruener, and Roland Siegwart. Probabilistic plane fitting in 3d and an application to robotic mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 927–932. IEEE, 2004.

[58] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[59] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.