ROYAL INSTITUTE
OF TECHNOLOGY

Fusion of Ladybug3 omnidirectional camera and Velodyne Lidar

Guanyi Zhao

Master of Science Thesis in Geodesy No. 3138
TRITA-GIT EX 15-011

School of Architecture and the Built Environment
Royal Institute of Technology (KTH)
Stockholm, Sweden

August 2015

Abstract

The advent of autonomous vehicles expedites the revolution of car industry. Volvo Car Corporation has an ambition of developing the next generation of autonomous vehicle. In the Volvo Car Corporation, Active Safety CAE group, enthusiastic engineers have initiated a series of relevant research to enhance the safety function for autonomous vehicle and this thesis work is also implemented at Active Safety CAE with their support.

Perception of vehicle plays a pivotal role in autonomous driving, therefore an idea of improving vision by fusing two different types of data from Velodyne HDL-64E S3 High Definition LiDAR Sensor and Ladybug3 camera respectively, is proposed.

This report presents the whole process of fusion of point clouds and image data. An experiment is implemented for collecting and synchronizing multi-sensor data streams by building a platform which supports the mounting of Velodyne, Ladybug 3 and their accessories, as well as the connection to GPS unit, laptop. Related software/programming environment for recording, synchronizing and storing data will also be mentioned.

Synchronization is mainly achieved by matching timestamps between different datasets. Creating log files for timestamps is the primary task in synchronization.

External Calibration between Velodyne and Ladybug3 camera for matching two different datasets correctly is the focus of this report. In the project, we will develop a semi-automatic calibration method with very little human intervention using a checkerboard for acquiring a small set of feature points from laser point cloud and image feature correspondences. Based on these correspondences, the displacement is computed. Using the computed result, the laser points are back-projected into the image. If the original and back-projected images are sufficiently consistent, then the transformation parameters can be accepted. Displacement between camera and laser scanner are estimated through two separate steps: first, we will estimate the pose for the checkerboard in image and get its depth information in camera coordinate system; and then a transformation relation between the camera and the laser scanner will be computed within three dimensional space.

Fusion of datasets will finally be done by combing color information from image and range information from point cloud together. Other applications related to data fusion will be developed as the support of future work.

In the end, a conclusion will be drawn. Possible improvements are also expected in future work. For example, better accuracy of calibration might be achieved with other methods and adding texture to cloud points will generate a more realistic model.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my gratitude to my KTH supervisor Milan Horemuz for giving me enlightening introduction and helping me get a comprehensive view of my thesis topic, offering me valuable suggestions on literature review and some constructive ideas, as well as the useful comments, remarks and engagement through the learning process of this master thesis.

Furthermore, I would like to give my sincere appreciation to my Volvo Car Corporation supervisor Patrik Andersson, who comes up with and proposed the idea of this thesis work, of his supports all the way on technical instruction, experiment preparation and knowledge supplement.

Also, I will give my genuine thankfulness to Yury Tarakanov, the second industrial supervisor from VOLVO CAR CORPORATION, who offers me the precious opportunity of doing thesis work in VOLVO CAR CORPORATION and gives me lots of help throughout my work.

Last but not least, I would like to thank all participants in my survey, who shares their precious time during the process of interviewing, and colleagues who enthusiastically offer me help in various aspects.

Chapter 1 - Introduction

Introduction will give a general description of the whole project from its background, tasks and related previous work.

1.1 Background

Research on driverless car has been conducted for years because of its advantages over human-controlled ones and possible applications on a wider range of field in long run. Even though it is still not mature enough to be viable due to the restrictions such as acceptance of technology and high cost, the future of driverless car is promising and the opportunity of owning an autonomous vehicle is coming.

A very obvious advantage of autonomous car over ordinary car is that it can reduce traffic accident dramatically. The majority of road crashes are caused by human error according to traffic research [43]. Autonomous car is without doubt a good solution to this serious road traffic problem. Besides, autonomous cars possess advantages such as reducing the possibility of unintentional moving violation, offering convenience for people who are unable to drive, liberating people from long-time mental stress on road and making the commute time productive.

The first autonomous car appeared in the 1980s. German pioneer Ernst Dickmanns got a Mercedes van to drive hundreds of highway miles autonomously [44]. After that more and more prototypes come into the world.

The original idea of this project comes from active safety group of Volvo Car Corporation which is highly interested in developing driver support functions. They have the ambition to develop the next generation of autonomous car for improving safety.

Autonomous vehicles should be capable of perceiving its surrounding environment and navigating with high accuracy, therefore sensors offering visual input are crucial components of them. In this project, we will design a workflow and develop related methods to carry out sensor fusion between Lidar and camera, to be able to colorize point cloud. Two available sensors Velodyne HDL-64E 3D-LIDAR scanner and Ladybug3 video camera with omnidirectional view will be employed in this work (see Figure 1-1 and Figure 1-2).

Both laser scanner and camera have their own advantages and disadvantages -

LIDARs are advanced sensors that record the surroundings of a vehicle as a three-dimensional point cloud, providing accurate shape and range measurements for surrounding objects. Moreover, in contrast to usual mapping approach with a single or few detailed scans using a terrestrial Lidar, the Velodyne Lidar can collect stream data containing large number of scans from 360 degree of view at different locations as the carrier vehicle moves.

Figure 1-1: Velodyne HDL-64E 3D-LIDAR scanner

Figure 1-2: Ladybug3 1394b

However, data obtained from the velodyne is very sparse, further interpolation might be necessary. Color and texture information, which are helpful for more detailed classification and simulation of real world objects, are unfortunately unavailable in point cloud. Besides, its 50 meters effective coverage range for pavement and 120 meters effective coverage range for cars and foliage might not be sufficient for autonomous vehicle to respond in some cases. Also, objects nearby will be invisible because of 26.8 degree vertical field of view limitation.

On the other hand, video cameras can shoot an area of much bigger radius. What's more, they are cheaper, more robust and provide color images with high resolution.

However, due to the lack of depth information, it is not straightforward to acquire original shapes, sizes orientation of various objects, which makes the detection of vehicles, pedestrians and other obstacles complicated.

Apparently, nice and complementary features of camera and LIDAR sensor make them each other's cooperative partner. More reliable and informative scenes will be possible to get through the fusion of data acquired from those two sensors.

Because of resource limit, we are currently unable to collect data in a traffic environment. Instead, we will focus on our methods of realizing the proposed idea, and perform a test in a more tractable condition with all available resource.

1.2 Objectives

To incorporate information from both sensors shooting in dynamic environment, the synchronization of both sensors is a prerequisite, which would ensure data acquired at the same time match perfectly. However, in this project, we will just synchronize different datasets off-line. Thus, instead of synchronizing hardware, 'soft' timestamps will be used for synchronizing datasets after data acquisition. In this step, GPS/IMU system is employed as an accurate timing apparatus.

Equipped with multiple complementary sensors, a 6-DOF rigid body transformation relating coordinates frames of different sensors is indispensable in order to represent sensed information in a common coordinate system. Therefore, Calibration is another preliminary step for data fusion. Only extrinsic calibration, or more plainly, inter-sensor calibration will be implemented in this case. We will mainly work on computing rotation and offset between laser scanner and camera.

A semi-automatic calibration method will be developed with a checkerboard for acquiring a small set of feature points from laser point cloud and image feature correspondences. Based on these correspondences, the displacement is computed. Using the computed result, the laser points are back-projected into the image. If the original and back-projected images are sufficiently consistent, then the transformation parameters can be accepted.

Velodyne HDL-64E S3 High Definition LiDAR Sensor and the high resolution Ladybug3 spherical digital video camera system are two primary sensors acquiring data from environment, both of which are fixed through a predesigned platform which is mounted on the top of a vehicle.

More specifically, we can break our objective into several pieces:

1) To set up logging and time-stamping for Ladybug3 360 degrees camera.
   Volvo Car Corporation has already solved Velodyne time-stamping problem, however setting up image acquisition and the image logging with GPS time-stamping needs to be established ("Soft" GPS time-stamping is used, i.e. the images are stamped when logged on the PC).
2) To develop a suitable method for calibrating offset between Lidar, camera. The calibration experiment shall be performed with help from VOLVO CAR CORPORATION, and the accuracy of result will be evaluated.
3) To develop a function for extraction of the part of the image corresponding to the bounding box of the given set of LIDAR points projected onto the image plane. Object and point positions will need to be interpolated to match the image acquisition time, using knowledge of their velocities and accelerations.
4) To develop a method to calculate and store colors for all Velodyne Lidar points.

A designed workflow clearly describes the above objectives:

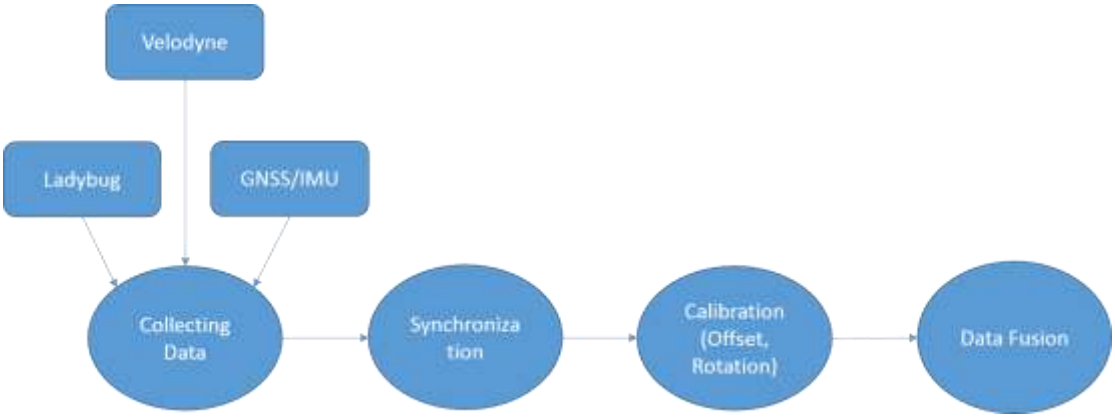

Figure 1-3: workflow of the project

1.3 Outline

Chapter one is an overview of the whole project. The rest of this thesis is organized as follows. In the next chapter, we give an introduction of important hardware and software as well as code resources that are used in this project. Chapter 3 presents details of performing an experiment to collect data and how we obtain data of the desired format. Chapter 4 states methods of the synchronization and calibration of sensors and fundamental mathematical support behind them. Experimental results with appraisal and explanation of them are given in Chapter 5, followed by a discussion in Chapter 6. Finally, a conclusion of the whole project and future work are presented in the last Chapter.

1.4 Related work

From the work breakdown, it is quite obvious that the synchronization and calibration of sensors are in fact the most essential work in this project. Lots of previous documents proposed or implemented similar work.

To start with, the experimental setup in [1] gives a good example of building hardware and software system, which is similar to our plan for mounting devices on vehicle. Another system containing a laser scanner, an infrared camera, a video camera and an inertial navigation system which is also designed to fuse data from the laser scanners and the infrared camera is presented in [2]. To understand GPS timestamp better, an introduction of NMEA 0183 which is an interface standard of GPS receiver is given in [3].

Various methods have been developed and implemented for sensors calibration. Calibration usually include two categories, intrinsic calibration or extrinsic calibration.

Intrinsic calibration of a sensor affects the way of data sampling, which is very important for the accuracy of acquired data and it also influences the extrinsic calibration. Extrinsic calibration indicates the rotation and offset of a sensor with respect to another coordinate system, which is usually performed for the purpose of relating the data of one sensor to those of the other, for example, boresight Calibration of camera/IMU systems [7]. It also could be implemented in order to relate the sensor data to those in real world.

Camera and laser range finder are two sensors commonly used in robotics. A large quantity of documents addressed problems of camera-laser range finer calibration in robotics application. They are also good choices for intelligent vehicles to recognize environment, we will calibrate camera and laser scanner for sensor fusion.

An intrinsic and extrinsic calibration of Velodyne laser scanner offers detail information of how laser beams related to the Velodyne coordinate system [5]. A geometric and photometric calibration of omnidirectional multi-camera is implemented [24], with a checkerboard and a total station. Paper [10] addresses the problem of estimating the intrinsic parameters of the 3D Velodyne lidar while at the same time computing its extrinsic calibration with respect to a rigidly connected camera.

Those articles present useful methods of calibration for the types of sensors we will use in this project. We will not cover all aspects given in them, e.g. intrinsic calibration will not be implemented since we can use intrinsic calibration parameters provided by manufacturer. Our intention is to map texture and color from image to point clouds, therefore only the displacement between them are of interest.

2D laser scanner is usually mounted on a robot for robot navigation. Many researches of calibration method between 2D laser scanner and camera were done.

An extrinsic calibration between camera and 2D laser range finder is implemented by first solving pose with respect to checkerboard and compute plane parameters, then using all laser points that lie on the checkerboard plane to add constraints to the transformation [8].

A few methods for estimating the relative position of a central catadioptric camera and a 2D laser range finder in order to obtain depth information in the panoramic image are presented in [22]. An algorithm for the extrinsic calibration of a perspective camera and an invisible 2D laser range finder is presented in [28].

2D laser scanners are used commonly for planar robotics navigation, but for restoring traffic environment, 3D laser scanner will provide definitely more detailed and important information.

The problem of 3D laser to camera calibration seems to be first addressed in a technical report which presented a Laser-Camera Calibration Toolbox (LCCT) [15]. The method is based on [8], but extended for the calibration of 3D laser scanner instead of 2D laser scanner. Another extrinsic calibration technique also developed from [8] but used 3D laser scanner and an omnidirectional camera system [9].

Extrinsic calibration between camera and laser data requires co-observable features in datasets from both sensors. Substantial prior papers addressed the problem of selecting common features for camera-laser scanner calibration.

Different techniques exist for the feature selection. Two categories of calibration methods: photogrammetric calibration and self-calibration are addressed based on whether we use an artificial calibration object or not [12].

For those methods that require calibration object, checkerboard is always chosen as the one because it is cheap to make, easy to employ, corners can be detected with high accuracy and point clouds also benefit from the regular plane feature.

Lots of calibration work employed checkerboard as the calibration target [8] [9] [27]. A fully automatic calibration method using multiple checkerboards placed at different locations so that all checkerboard are included in one single shot is proposed [25].

Paper [12] also requires the camera to observe a planar pattern shown at a few (at least two) different orientations. This paper claims that the approach lies between the photogrammetric calibration and self-calibration because 2D metric information are used rather than 3D or purely implicit one.

There are also articles using self-calibration method.

A camera-3D laser calibration method which does not require any calibration object but use a few point correspondences is presented [14]. This technique is used for the calibration of a 3D laser scanner and omnidirectional camera through manual selection of point correspondences between an image and its corresponding range image.

Automatic calibration without target, such as the mutual information (MI) based algorithm for automatic extrinsic calibration of a 3D laser scanner and optical camera system [11] [13]. The mutual information method is intelligent and suitable for in-field work however it requires either close objects or multiple scans according to their result and analysis [11]. Besides, our noisy data make this method difficult to implement. In our case, using classic approach for calibration is preferable.

Lots of previous studies used well-designed calibration instruments or environmental features to improve the calibration accuracy. However, the performance of those methods largely relies on the quality of laser point clouds, such as point density and the actual location of scanned points on the calibration object or the environmental features.

Because of the low resolution of Velodyne, it is difficult to get trustworthy individual target points. To solve this, taking advantage of some geometric objects such as plane can be a solution. A polygonal planar board method is used for this purpose, by estimating the vertices of polygonal board from the scanned laser data and using the same set of points in 2D image to apply a point to point correspondences for calibration [6].

When solving calibration parameters, non-linear optimization method is usually used to improve accuracy. Lots of existing approaches solve this nonlinear estimation problem through iterative minimization of nonlinear cost functions. Those non-linear optimization methods usually require initial estimate which is acquired from linear solution, the accuracy of the final result hinges on the availability of its precision.

Some previously mentioned articles also did this work. A method which employs the Nelder–Mead direct search algorithm to minimize the sum of squared errors between the points in images coordinate system and the re-projected laser data by iteratively adjusting and improving the calibration parameters is presented in [4].

Nonlinear optimization using Levenberg–Marquardt algorithm to minimize the combination of re-projection error and laser to calibration plane error is implemented in [8]. An initial guess is acquired from linear solution before optimization. Calibration method presented in [24] also targets at minimizing projected position of a measured 3D position and the detected 2D position, 3D position is measured by total station.

Paper [25] presents a method which implements an initial transformation first using plane to surfaces alignment based on centroids and normal vectors, then a fine registration using an iterative closest point optimization to minimize the sum of points to point distances.

Pose estimation or PnP (Perspective n Points) has been researched for long time in computer vision. We will also cover this field when we try to correspond our image to the real objects in world. Further discussion is available in section 4.3.3.

Chapter 2 - Overview of involved tools

A general knowledge about hardware and software we use in this project is necessary since some features of them are important for our further work presented in other chapters.

Ladybug3 camera and Velodyne laser scanner are sensors that perceive and generate data of environment. GPS/INS box is the synchronization tool of Ladybug3 and Velodyne. Laptop will record and process all collected data. In this chapter, we will give a description of important information of them.

2.1 Ladybug3 camera

Ladybug3 is the sensor we use to collect image. Data will be stored in the form of stream. As a complete package, Ladybug3 camera and its enclosed software components play an important role in data collection, data processing and even camera calibration. Therefore, an insight of Ladybug 3 is indispensable for the following work. In this section, a comprehensive introduction will cover all must know facts of Ladybug 3.

2.1.1 General description

The Ladybug3 spherical digital video camera system produced by Point Grey Research (PGR) includes both hardware (a Ladybug3 camera and all the necessary hardware to get the camera running) and software (a license of the Ladybug software development kit).

Ladybug3 camera is composed of six high quality 2-Megapixel (1600x1200) Sony image CCD sensors, among which five sensors are positioned in a horizontal ring and one is positioned vertically. All the sensors can generate up to 12 million effective pixels in total. This design of camera enable it to cover more than 80 percent of a full sphere and offer images of high resolution.

One can visit its official website [17] for more detailed information.

2.1.2 Ladybug coordinate system and intrinsic calibration

Most of our data processing and calibration work require knowledge about how Ladybug3 works and how we operate it to get the desired output, especially its coordinate system.

There are a total of seven 3D coordinate systems and six 2D pixel-grid coordinate systems on every Ladybug camera. That's because each lens has its own right-handed 3D coordinate system as well as a 2D pixel-grid coordinate system. In addition, there is a Ladybug 3D Coordinate system that is associated with all the cameras as a whole.

In 2D coordinate system, the u and v axes are the image based 2D image coordinate system for the rectified image space. 2D pixel location of raw image is easily rectified using Ladybug API functions.

In the coordinate system, the origin is at the intersection of the optical axis and the rectified image plane; u axis points along the rows of the image sensor in the direction of ascending column number (i.e. to the right) and v axis points along the columns in the direction of ascending row number (i.e. down). Points are measured in pixels in 2D coordinate system.

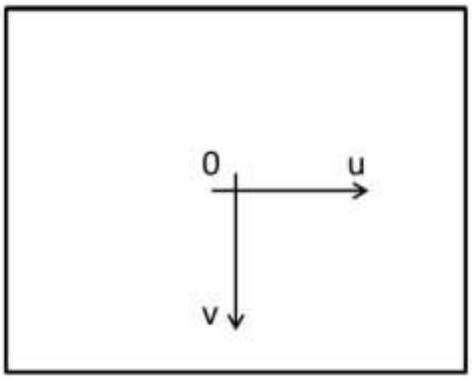

Figure 2-1: 2D coordinate system of camera lens

In the individual lens 3D coordinate system, the origin is the optical center of the lens; Z axis points out of the sensor along optical axis. The X and Y axis are based on 2D image coordinate system. The Y axis points along the image columns, which corresponds to v, and the X axis points along the image rows, which corresponds to u. Different from 2D coordinate system, units are meters, not pixels.

The Ladybug Camera coordinate system is centered within the Ladybug case and is determined by the position of the 6 lens coordinate systems [17]:

- Origin is the center of the five horizontal camera origins
- Z axis is parallel to the optical axis of the top lens (lens 5)
- X axis is parallel to the optical axis of lens 0
- Y axis is consistent with a right-handed coordinate system based on the X and Z axis

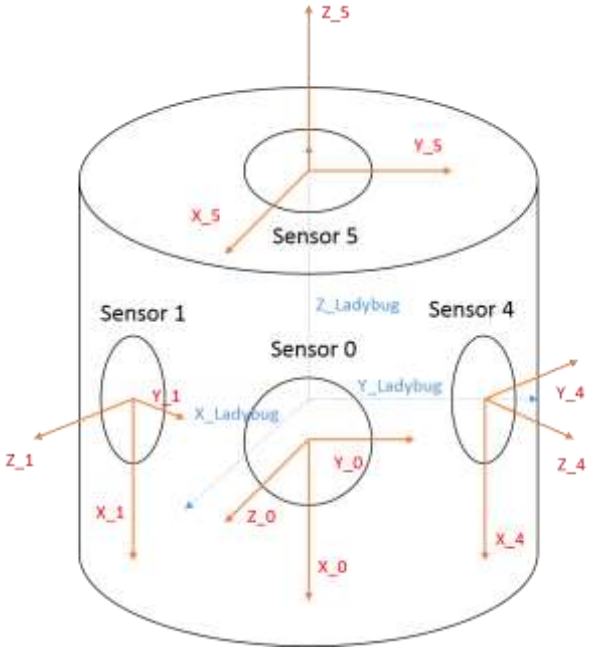

Figure 2-2: 3D coordinate system of Ladybug and each lens

Accurate calibration is a guarantee for effective warping and stitching of the images produced by the camera system's six sensors. It will come out good result by applying the sophisticated calibration of all sensors and the distortion model of their lens.

All cameras are pre-calibrated by the manufacturer and necessary intrinsic parameters (e.g. focal length, the rectified image center) of each individual camera are well known. Those parameters could be utilized directly for image processing or for other purposes. For instance, computation of camera position and orientation will use intrinsic parameters inevitably. In camera calibration, it is necessary to find the corresponding pixel in an image for a given point of 3D coordinates. By using the focal length and image center, we can apply the normal projection equation to the 3D point (which must be in local camera frame) to determine where it falls on the ideal image plane.

Meanwhile, the extrinsic transformations between the different lens coordinate system and the Ladybug camera system are provided by the manufacturer with high accuracy as well. Therefore, the measurements from any of the cameras can be easily transformed to the Ladybug's fixed frame of reference.

2.1.3 Ladybug Software Development Kit (SDK)

The Ladybug3 performs all the image acquisition, processing, stitching and correction necessary to integrate multiple camera images into full resolution digital spherical and panoramic videos in real time. Users are open to choices of outputting different types of images by using Ladybug SDK. If anyone wants to connect a GPS receiver to a laptop and insert NMEA data into Ladybug images, the GPS receiver should have a serial or USB interface for connection and be able to stream NMEA data in real time [16]. In order to implement those functions through user's control, a software system is necessary.

In fact, aside from hardware, the Ladybug3 spherical digital video camera system includes a feature rich Software Development Kit (SDK) to manage image acquisition, spherical and panoramic image production, and camera settings. The Ladybug SDK is a software package composed of a device driver, Application Programming Interface (API) software library which allows users to integrate Ladybug functionality with custom applications, variety of example programs and affiliated source code, the LadybugCapPro application which allows users to control many camera functions without any additional programming.

2.1.3.1 LadybugCapPro program

Two programs are available for controlling the Ladybug3: the LadybugCap and LadybugCapPro programs. LadybugCap offers basic functions [17] such as

- View a live video stream from the camera.
- Display fully stitched panoramic and spherical images.
- Save individual panoramic images or stream files.
- Adjust frame rates, properties and settings of the camera
- Access camera registers.

The LadybugCapPro application contains more comprehensive functionalities than that of LadybugCap. It can be used in conjunction with a GPS device, which can then be recorded into a stream in addition to functions that are incorporated by LadybugCap.

In our first experimental data collection, we will use LadybugCapPro to collect images and record GPS data since it is easy to operate and our own program is not fully completed at the beginning. In fact,

API (will be introduced in the following section) offers opportunities for users to design their custom programs for collecting data and it is expected that everything could be done by designing a customized application. We will go for that after all the other work is accomplished. Hopefully, we will execute our code for data collection in the end.

2.1.3.2 Application Programming Interface (API) software library

PGR Ladybug includes a full Application Programming Interface that allows customers to create custom applications to control Point Grey spherical vision products.

Even though LadybugCapPro is a well-designed and handy software working pretty good for Ladybug camera, a customized application is preferable for some specific usages according to different user's demands.

Ladybug API makes it much easier for users to create their own customized applications. With API software library, one can program multifarious functions flexibly and make their own desirable applications.

Since we intend to make our own applications, Ladybug API makes big contributions in image processing related programs.

2.1.3.3 Examples in the C/C++ programming environment

The SDK provides a number of sample programs and source code as perfect instructions for programmers when exploring API software library.

Examples range from simple, basic programs such as LadybugSimpleGrab, which illustrates the basics of acquiring an image from a Ladybug camera, and LadybugSimpleGPS, which shows how to use a GPS device in conjunction with a Ladybug camera to integrate GPS data with Ladybug images to more advanced examples which demonstrate complex functionality.

Users can feel free to develop their own program by enriching the existing sample code. In this project, for example, one application developing from a sample code file called 'LadybugSimpleRecording' is used as data collection tool. As we mentioned before, in the first experiment, we simply use LadybugCapPro to collect data, however, more interesting tasks could be achieved flexibly by customizing 'LadybugSimpleRecording' in the way we desire, which makes it a better option for data collection. Another important example that used in our project is the stream processing application which is extended from a sample file called 'LadybugProcessStream' to access images in stream files and output them as different formats (BMP, JPEG) and types (raw image, rectified image, panoramic image, spherical image, dome projection image).

2.2 Velodyne HDL-64E S3 High Definition LiDAR Sensor

In this project, not so much work will be performed on Velodyne LiDAR Sensor since Volvo Car Corporation has been researching and using it for long time and they have a good knowledge about it. The content of this section mainly tells about physical structure of Velodyne and its working mechanism.

2.2.1 General description

Velodyne HDL-64E S3 High Definition LiDAR Sensor contains 64 lasers, of which 32 lasers are mounted on upper and the other 32 lasers are mounted on lower laser blocks. All lasers from those two blocks rotate together as one entity although they are mounted separately. The spinning rate of sensor ranges from 0300 RPM (5 Hz) to 1200 RPM (20 Hz). The default is 0600 RPM (10 Hz).

The sensor covers a 360 degree horizontal Field of View (FOV) and a 26.8 degree (-24.8° down to +2° up) vertical FOV. Angular resolution of the scanner are: azimuth angular resolution 0.09 degree,

vertical angular resolution approximately 0.4 degree. All the parameters can be found in the manual guide of Velodyne HDL‐64E S3 [18].

2.2.2 Velodyne Coordinates System and calibration

Cartesian coordinates of 3D point are determined by measured distance, current rotational (horizontal) angle of laser and vertical rotational angle, which is fixed for each laser, and correction parameters as vertical and horizontal offset. Distance standard error of Velodyne HDL-64E-S3 in datasheet is determined < 2cm.

Scanner coordinate system is right-handed and orthogonal. Origin is in the center of the base [20].

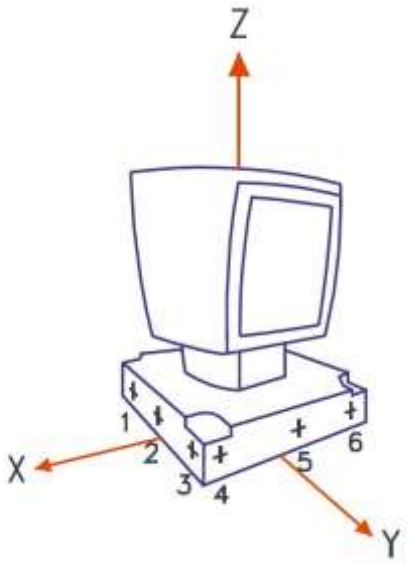

Figure 2-3: Coordinate system of Velodyne laser scanner (image from [20])

The sensor needs no configuration, calibration, or other setup to begin producing viewable data. Once the unit is mounted and wired, supplying power to the sensor will cause it to start scanning and producing data packets.

However, calibration file is required to set the accuracy of the data. One can create a calibration table either from the calibration data included in-stream from the sensor or from the.xml data file included with the sensor [38]. In our project, we choose .xml file for calibration raw data points. The used calibration file is offered by Volvo Car Corporation.

2.3 GPS

A satellite navigation system with global coverage is termed a global navigation satellite system (GNSS). GNSS is a satellite system that is used to pinpoint the geographic location of a user's receiver anywhere in the world. The term GPS (Global Positioning System) is specific to the United States' GNSS system, which is a space-based satellite navigation system that provides location and time information on or near the earth where there is signal from four or more GPS satellites.

The GPS system is primarily intended for navigation and positioning. However it enables ground-based receivers to obtain accurate timing information as well with precise atomic clock installed on satellites. A timing receiver can provide a consistent supply of accurate time stamps to a host computer with a well-located GPS antenna.

The positioning function is of interest as well in further applications, for example, it might be necessary to interpolate moving object and point positions in order to match the image acquisition

time, using knowledge of their velocities and accelerations. Nevertheless, a primary task is to get timestamps from GPS and insert them to stream files.

2.3.1 RT3000 GNSS/INS and Garmin 18x LVC GPS Unit

There are two types of GPS used in this project – one is RT3000 Inertial and GPS Navigation System, the other is Garmin 18x LVC GPS Unit.

When sensors are collecting data outdoors, RT3000 Inertial and GPS Navigation System is chosen for data collection because of its good features such as high accuracy of position and orientation and high frequency of data output. Before the RT3000 can start to output all the navigation measurements it needs to initialize itself. It is necessary to drive the vehicle during the first 15 minutes of operation, otherwise the errors will not be estimated and the specification of the system will not be reached [36].

For the indoors experiment we will use Garmin 18x LVC GPS Unit which includes GPS receiver and antenna since all sensors will remain static and the small size of Garmin GPS unit is easy to carry [37].

2.3.2 Data Format

The data that can be received and transmitted by the GPS receiver is a series of standard "sentences" that contain data including information of time and date, geographic position - latitude and longitude, and individual satellite, which are defined by the NMEA (National Marine Electronics Association standard). NMEA standard is accepted by nearly all the GPS device, and Garmin 18x LVC GPS Unit is not an exception.

There are different NMEA sentence types, GPGGA (Global Positioning System Fix Data), GPZDA (Date & Time), GPRMC (Recommended minimum specific GPS/Transit data) and so on. If we use GPRMC data which contains the recommended minimum data for GPS, date and time information will be acquired from its members: ucRMCHour which indicates hour (Coordinatesd Universal Time), ucRMCMinute (minute), ucRMCSecond (second) and wRMCSubSecond (hundredth of a second).

It is the time information that matters in this project and the accuracy of it is pivotal. Timing data is transferred in NMEA messages, sometimes additionally PPS pulse is provided by “better” GPS units. A pulse per second (PPS or 1PPS) is an electrical signal that has a width of less than one second and a sharply rising or abruptly falling edge that accurately repeats once per second, which is used for precise timekeeping and time measurement.

NMEA without PPS doesn’t give accurate synchronization (for reasons like: delay before sending NMA from GPS receiver, time calculation inaccuracies due to few satellites, atmosphere…).

According to the materials we search, only RS-232 serial port support PPS output.

2.3.3 Connection to computer

RS232 serial interfaces are a preferred means of communicating timing information between references and PC’s. Serial interfaces consist of data transmission lines and control lines. The data transmission lines convey actual character-based time and date information from device to a host PC.

Due to buffering of characters, the data lines alone cannot be used to convey accurate time. However, RS232 input control lines are connected to hardware interrupts and can be used to provide highly accurate event timing. The control lines indicate status information, such as ‘Clear to Send’ (CTS) and ‘Ready to Send’ (RTS). Using the data transmission lines in conjunction with the control lines, driver software installed on the computer can then read time stamps from the receiver and very precisely adjust the operating systems time so that it coincides with the correct time.

USB interfaces can also be used for timing but are essentially software based and have an inherent processing delay. This inevitably means that they are not as accurate as RS232 serial interfaces for timing purposes.

2.4 Programming software and important libraries

We will design a package of applications through programming in order to process data, test our methods and get the final results. This section briefly present those useful programming tools and resource we use in this project.

2.4.1 Programming software

Most of the methods will be implemented through coding in C++ programming environment. Microsoft Visual C++ 2010 is the programming tool we use. Meanwhile, Matlab will be used as assistant tool sometimes for its advantages on matrix manipulations, plotting of functions and data as well as its big collection of various toolboxes.

2.4.2 Libraries

PCL and OpenCV are two main libraries we use aside from above-mentioned Ladybug API library.

The Point Cloud Library (PCL) is a large scale, open-source library. It is widely used for point cloud processing and 3D geometry processing.

Various algorithms and examples written in C++, for feature estimation, data filtering, surface reconstruction, registration, model fitting, recognition, visualization, segmentation and some more advanced applications, are available in the library.

Open Source Computer Vision (OpenCV) is another big open-source library we use in this project. It contains programming functions mainly aiming at real-time computer vision.

Chapter 3 - Data acquisition and processing

We first need to design an experiment to collect data. An original plan was to drive around and collect data in a traffic environment and the collected data are supposed to resemble to that provided by 'KITTI Vision Benchmark Suite' [24]. Unfortunately, a platform for mounting all sensors has not been completely built yet hence we temporarily are unable to drive to collect data outdoors.

To ensure that all sensors can be mounted stably without the presumed platform, instead of placing sensors on a moving object, a static environment is a more appropriate substitute for the setup of sensors.

The static environment makes synchronization a seemingly unnecessary step since everything is still. However, we do want to go through the whole plan as it is proposed in the beginning without skipping any important procedure. In fact, if we can manage to insert timestamps into data in the designed experiment, it will work equally well when data are collected in traffic environment.

This experiment is performed in a room. Our main intention is to test all devices and collect data for sensors synchronization and calibration. Ladybug3 camera and Velodyne combined with GPS unit will collect data continuously and finally generate Ladybug stream files and Velodyne PCAP files with GPS timestamps. Further processing work will enable us to extract data of desired format from image stream file and PCAP.

Matching data both spatially and temporally are significant. GPS can solve the synchronization problem by inserting timestamps. Camera is placed at an arbitrary angle and distance with respect to the laser scanner. Position and orientation of the two sensors are different on the platform, therefore we need to compute rotation angle and offset between these two sensors in order to express all data in the same coordinate system.

For calibration only a few pictures and scanned scenes will meet our demand, stream data are recorded for synchronization even though it is unnecessary under this circumstance. Basically, the rest of work will be based on the data we get in this experiment.

It is important to get familiar with sensors. Laser scanner is not new to us since Volvo Car Corporation are proficient at operating Velodyne and processing data acquired from it. Unlike Velodyne, the Ladybug 3 is a completely new device to us and it is the main sensor that we research on. It does take time to Figure out how to use Ladybug API to record and read data as well as build connection with GPS and insert timestamp into data.

If the synchronization and calibration results turn out to be good, we can just repeat the calibration procedure after the platform is well built and move the platform on the top of a car for data collection in traffic environment.

In order to control camera recording process, one can either use the enclosed software LadybugCapPro or writing a program with Ladybug API. LadybugCapPro offers user friendly interface to help to read stream file and output images. However, using C++ to program a custom application seems to be more desirable when it comes to operation efficiency and demand of certain functions.

Therefore, to implement both data acquisition and data processing, we prefer to develop our own independent custom programs for more flexibility.

3.1 Data Acquisition

It will take some time to build a new platform which can hold both Velodyne and Ladybug3 fixedly. There are actually lots of existing prototypes, and one similar setup can be found in 'KITTI Vision Benchmark Suite'. Once the platform is built, we can put it on a vehicle and collect data with relative position and orientation between sensors unchanged. With such a platform, calibration can be done in any place since placement of different sensors will not be easily disrupted.

In this project, a temporary system of sensors is built by simply mounting Velodyne to a pre-designed platform and putting Ladybug camera on the floor which just stands in front of Velodyne. Because of the limitation of experimental tools at present, Ladybug3 is unattached to the platform of Velodyne. We build this tentative system by placing sensors together in an appropriate way, which is really expedient without costing so much manual work but can still achieve the goal of testing and calibrating sensors.

All sensors will be connected to a laptop through which we can control data acquisition in real time.

GPS unit is connected to the laptop as well and writing timestamp to data from both sensors. RS232 is used to receive GPS data since we want to keep it as accurate as possible. To make our GPS work, attention should be paid on certain parameters. Baud rate of receiving and transmitting data should be set to the same value. We also need to choose the correct com port and avoid conflict with other programs.

Figure 3-1: Setup of sensors in the experiment

Velodyne and Ladybug sensors will capture data by scanning and shooting the room from the omnidirectional view. Related programs will run at the same time to output and store those data.

A very mature and popular camera calibration method is observing an artificial calibration object of precisely known geometry in 3-D space. Usually a classical checkerboard is employed as the calibration object. Sometimes, an elaborate instrument comprised of several orthogonal planes is set up for calibration, nevertheless it is not time efficient and cost expensive manual work.

In our experiment, only one checkerboard is employed. Theoretically, one image is enough for computing the calibration parameters. However, as it is introduced in Chapter 2, Ladybug3 get the

omnidirectional view with 6 camera sensors, five of which (horizontal ones) have overlapping view with Velodyne. Calibration will achieve higher accuracy by taking advantage of the omnidirectional view instead of using only a partial view. Therefore, we should move the checkerboard around let all five horizontal sensors detect the calibration object instead of using only one sensor.

However, Ladybug 3 is surrounded by the platform of Velodyne. Sensor 2 and sensor3 (Figure 3-1) of Ladybug3 are blocked by obstacles (Figure 3-2), therefore only three sensors are capable of acquiring the calibration object in an image.

In this experiment, the checkerboard is placed at three different positions where checkerboard can be detected by Ladybug sensor 0, sensor 1 and sensor 4. We place the checkerboard at a certain angle towards wall to make it more distinguishable from walls for plane segmentation in later work. The checkerboard is firstly placed in a corner which sensor 0 of Ladybug3 is facing to, as Figure 3-2 display. Next, checkerboard will be put in other two different positions (where sensor 1 and sensor 4 can reach the board) subsequently in the room.

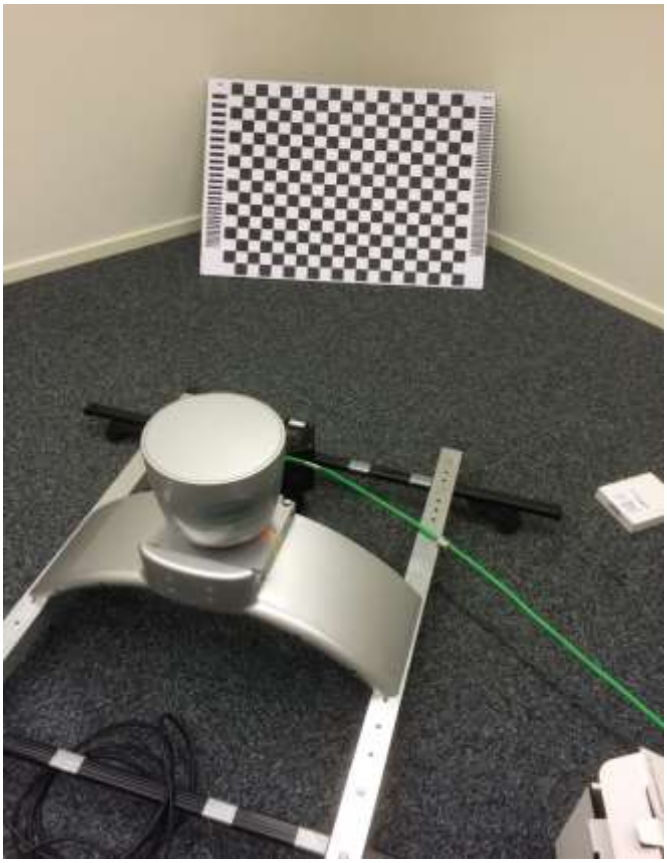

Figure 3-2: Setup of sensors in the experiment

A problem is that different sensors has its own coordinate system. Since the checkerboard is shot by different camera sensors (sensor 0, sensor 1 and sensor 4), rotation and offset among those camera sensors should also be taken into consideration. For the simplicity of extrinsic camera calibration, one can solve this problem by expressing all coordinates of points in the head frame of Ladybug3 for calibration.

3.2 Data Processing

Ladybug stream file is a collection of images and Velodyne PCAP file is a collection of scans, which are the two major data sets we will need. Besides, GPS timestamp extracted from both datasets are very important.

3.2.1 Output image

To extract separate images from stream file, we will develop a stream processing program with all necessary functions offered by Ladybug API.

This stream processing program makes it easy to handle the stream files. From reading, processing to outputting and saving images, stream processing covers most of the important functions to acquire image-related information.

It allows users to output the images as various types including raw image, rectified image, panoramic image, spherical image and dome projection image, as well as of different formats such as BMP, JPEG, TIFF and PNG to meet different requirements of further image processing. It can either read images from the very first image of a stream file or go directly to the number of image user specified. In addition, it can also extract GPS timestamp information.

Raw images and panoramic images are the two types we use later (see appendix A to find those images). Raw image are the input for checkerboard corner detection and panoramic image will be the file which we extract color information from. The size of raw images are 1616x1216 pixels and resolution of panoramic images are 2048x1024 pixels.

3.2.2 PCAP to PCD

Similar to stream processing, we need to get the scans in the form of point cloud data. An application ‘pcap2pcd’ will complete this task. It is offered by Volvo Car Corporation, author will use it directly with the permission of VOLVO CAR CORPORATION. A calibration file of Velodyne from manufacturer is used to calibrate raw points.

Point clouds data we get is very noisy (check appendix A), which brings severe interference to a series data processing work, such as object segmentation and visualization.

We temporarily will not spend much time on figuring out a good method to remove noise from point clouds, instead, we will try to use this imperfect data to test our calibration method. If we can get an acceptable result from the noisy data, we will continue work on data refining in the future to get more reliable point clouds.

3.2.3 GPS timestamp

Reading timestamp from cloud points has been previously done by Volvo Car Corporation. Now we just need to get GPS timestamp from images.

The format of GPS NMEA sentence is addressed in section 2.3.2. Time information should include hour, minute, second and sub-second (hundredth of a second). It is important to ensure timestamp include sub-second information which will help to synchronize datasets accurately.

To get the GPS data for the specified NMEA sentences from a Ladybug image, one can use the function call ‘LadybugGetGPSNMEADataFromImage’, from Ladybug API.

3.3 Data quality

The experiment is not so rigorously implemented because of limited time and resource. Thus, the interference from various noise will add limitation and difficulty to the calibration work.

The most serious problem is the noisy point clouds. We temporarily have no effective method to eliminate influences caused by the noise. Segmentation algorithm suffers from this problem even though we filter the point clouds and apply optimization method to segment the plane from its background.

We also find that the color (black and white pattern) has impact on the scanned points. Points reflected from black square pattern are a little bit further away than points reflected from white square pattern in laser scanner coordinate system. This is because white color returns stronger signals therefore less response time is required.

Other objects are very chaotic and barely discernible in point clouds. The final colored point cloud of the scanned room appears a bit confusing because of this. Unfortunately no other referential object could be used to verify the sensor fusion result, and checkerboard is the only distinguishable object for observing the final mapping result.

Images are of very good quality. However, as it is mentioned above, the view of two sensors is blocked by the arch and beams of Velodyne platform. If one can make the most of omnidirectional view of Ladybug3 by using all five horizontal sensors to acquire board pattern, a more balanced input will be used for calibration.

In fact, the view of sensor 1 and sensor 4 is also partially blocked by beams. This makes the colored point clouds not completely consistent with that in mage.

For the GPS data, hour, minute and second information are recorded but the sub-second information is unavailable. According to Volvo Car Corporation, the RT3000 they use can output sub-second information. It might be that The Garmin used in this experiment is incapable of providing sub-second information. Further test will be implemented in the future.

Chapter 4 - Methodology

In this chapter, we will present some methods that we use to implement the sensors synchronization and external calibration.

We develop a program for recording and logging timestamps for synchronization of data. Synchronization can be easily achieved by matching timestamps from both datasets as long as we get reliable and sufficiently accurate timestamps.

To implement the calibration, several methods are proposed. Because of the poor quality of the point clouds, some elaborate calibration methods are inappropriate to apply. We will implement external calibration for sensors using a very simple semi-automatic calibration method.

Calibration object will be detected automatically in both image and point cloud, followed by a point correspondences selection. Automatic target detection can avoid error caused by human and meanwhile reduce manual work. The so-called 'manual selection' of points only require someone to specify points to be used for calibration from those automatically detected points, thus the value of those points will not be influenced by human.

Evaluation methods include a comparison with ground truth which only offer an approximate reference, visual check and statistics of transformation error. We mainly address how we get the ground truth in this chapter, other evaluation method will be presented together with results.

4.1 Synchronization

The focus of the synchronization work in this project, is making a log of timestamps for Ladybug3 360 degrees camera.

In the hardware part, it takes a bit effort to find laptop with comport and figure out the output from GPS receiver. We first need to finish installation and connection of all necessary devices, then we check if there is any problem of receiving data from GPS or transmitting NMEA data to PC. Once all those work are done, it is not a challenging task of reading timestamp and synchronizing data sets.

During the above-mentioned experiment, the timestamp is stored in the stream file and PCAP file, we can get the timestamp when we read the images and point cloud data.

The basic idea of this synchronization is to insert timestamps into data and read them afterwards. We do not synchronize sensors in real time but develop an offline method of synchronization.

In real traffic environment, one should make sure the data will be matched accurately. Data will be fused incorrectly if we map an image into a point cloud which cannot correspond to the image. Both video camera and laser scanner can collect data at very high frequency. Therefore, millisecond accuracy should be provided for time-stamping. As we mentioned before, no sub-second information is available from the small Garmin GPS receiver. RT3000 GPS/IMU system will be employed in replace when we drive around to collect data.

This experiment will only capture data in a room, so it is unnecessary to really match the timestamps for the data. In this case, it is also impossible to verify the synchronization result by visual inspection and it makes no sense to match timestamp of only second accuracy. We do not have meaningful data to present for this part of the work, but we believe this timestamp logging method is feasible since a similar synchronization method has been practiced in other previous work, for example, KITTI Vision Benchmark Suite offers such timestamp logs in datasets.

4.2 Calibration

A semi-automatic calibration method with very little human intervention is performed using the technology of computer vision and point clouds processing algorithms.

In order to map the image information from the environment onto the range information precisely for creating realistic virtual models, we need both data sets expressed in one common coordinate system. To accomplish this task, extrinsically calibration between camera and 3D laser scanner must be implemented. Accurate calibration is a guarantee for the accuracy of sensor fusion.

Extrinsic calibration between sensors will be practiced to compute transformation parameters which are composed of rotation and offset between different coordinate system.

Calibration method can be classified into several categories depending on the how many dimensions of geometry are available - 3D reference object based calibration, 2D plane based calibration, 1D line based calibration and self-calibration. It is recommended that using a 3D instrument to calibration camera when accuracy is prioritized in a task because highest accuracy can usually be obtained by using a 3D information [42].

In order to achieve high accuracy, we will use the method of finding common points with known coordinates in both sensors' coordinate system to compute extrinsic calibration parameters. Usually, features which could be easily recognized and detected will be preferred such as intersection of lines.

In the experiment, we employ a checkerboard of regular and noticeable pattern as our calibration target. In fact, based on the quality of acquired laser points, checkerboard is the only recognizable target we can use. Those coordinates of corner points of checkerboard squares could be easily detected from image and computed from point clouds. We will get those coordinates in image through corner detection and those from point clouds through board extraction and a bit extra computation work.

We will first present how to find the point correspondences, including a corner detection algorithm in image, and a board extraction algorithm with explanation about how to select corner points of squares in point clouds. Figure 4-1 is a description of the calibration work flow. In the figure, P_i, P_j, P_k represent the chosen point correspondences for calibration.

Next, a basic geometric camera calibration model will be reviewed, which represents the relation between 2D points on image and their pose in 3D world. Subsequently, a method for pose estimation which is also known as the classical perspective n point (PnP) problem will be presented as well as the solution to the problem. The output of the pose estimation is the depth of those point.

Finally, it is the time to compute parameters for 3-D rigid body transformation that aligns two sets of points using known correspondences. This transformation is implemented within 3D space, therefore we do not need camera projection model in this step.

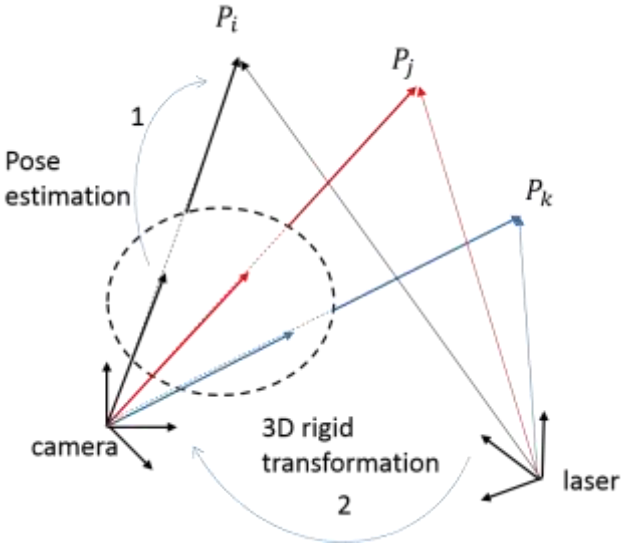

Figure 4-1: Calibration in two steps

4.2.1 Points correspondences

To select the corner points manually from image is easy to implement, however an automatic corner detection will help to reduce errors caused by human and elevate efficiency.

One of the earliest corner detection algorithms is the Moravec corner detection algorithm. Later Harris Corner Detector improved Moavec coener detection and it is widely used. The Shi-Tomasi Corner Detector [26], which is the corner detection method we use for corner detection, develops from Harris Corner Detector but works much better.

OpenCV (Open Source Computer Vision) offers user various corner detection functions, we will use the existing functions directly from library to detect corners. The corner detection algorithm will find all square corners on checkerboard, which are much more than we need. Therefore, an interaction function is designed for manually selecting points. In this case, one just need to click on the detected points to specify the ones used as points correspondences. No human error will be introduced since all points are detected automatically and the accuracy of their value only rely on corner detection algorithm.

Unlike images, it is impossible to either select corner points manually or detect those points automatically in point clouds since board pattern is chaotic and no clear feature could be used for automatic extraction. Therefore, we need to compute coordinates for those corner points mathematically in combination of known checkerboard measurements.

We will find all corners through following steps:

1) Extract board from point clouds;
2) Detect the contour of board and vertices of hull;
3) Compute the coordinates for every square corner.

This board extraction algorithm is basically derived from examples presented in PCL. In step one, segmentation work uses resource from the ‘pcl_sample_consensus’ library, which is included in PCL. It holds SAmple Consensus (SAC) methods like RANSAC (RANdom SAmple Consensus) [40] and models like planes and cylinders. Methods and models can be combined freely in order to detect specific models and their parameters in point clouds [21]. For example, SAC_RANSAC, which can deal with outliers and build mathematical model through iterative computation from noisy data, is the

segmentation method we use while SACMODEL_PLANE is the model type we will use to extract board since it is used to determine plane models. Four coefficients of the plane, representing its Hessian Normal form (a, b, c, d) will be returned. RANSAC is known for its excellent ability of robust estimation when dealing with outliers and it is commonly used in various segmentation work.

Before the board extraction, using a 'PassThrough' filter can delete points that are either inside or outside a given range along a specified dimension. Similarly, the 'VoxelGrid' reduces the data by assembling a 3D local voxel grid over the given point cloud data and approximating with their centroid. Both two filtering method will cut down point cloud size and improve segmentation efficiency. Introduction and application of these two filtering method are available in PCL too.

Next, construct a convex hull for the extracted board and slit it into four lines. Then we will get its vertices by computing the intersection of those crossing lines.

Finally, coordinates of all points could be easily computed:

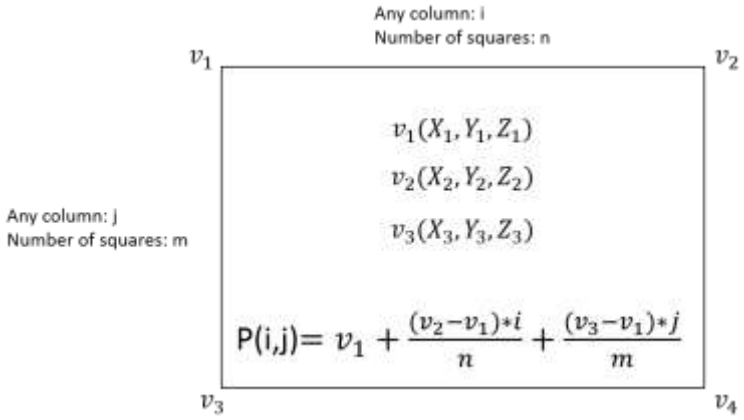

Figure 4-2: Compute square corners of chess board extracted from point cloud

4.2.2 Geometric camera calibration model

Geometric camera calibration is required for extracting metric information from 2D images and relating points on image to points in other 3D coordinate systems. Geometric camera calibration includes the acquisition of camera intrinsic and extrinsic parameters and pinhole camera model is applied in this case.

Intrinsic camera parameters encompasses focal length, lens distortion parameters, center of distortion and aspect ratio, among which nonlinear intrinsic parameters such as lens distortion are estimated to rectify a distorted raw image. Linear intrinsic parameters consists of focal length, image sensor format, and principal point. Those linear intrinsic parameters will be the parameters estimated in pinhole camera model.

In fact, Ladybug3 SDK contains perfect API to rectify image, therefore it is unnecessary to estimate those intrinsic camera parameters once more. Those parameters could be treated as known conditions and used for further applications.

Extrinsic parameters indicate the rotation and translation, which are termed a 3D rigid transformation, between 3D camera coordinate system and other coordinate system.

The camera calibration is a combination of a rigid body transformation (3D world coordinates to 3D camera coordinates of each object point) and perspective projection (3D image scene to 2D image) using Pinhole model. One can relate the camera 2D coordinates (pixels) of a projected point to the 3D

coordinates of its counterpart in defined coordinate system after the process. The equation (1) represents the calibration process clearly [41].

$$\begin{bmatrix}\text{2D image point (u, v)}\\ \text{in homogeneous}\\ \text{coordnate system}\end{bmatrix}_{3X1} = \begin{bmatrix}\text{3D to 2D perspective transformation}\\ \text{from object space to image plane}\\ \text{using object to image projective}\\ \text{transformation (intrinsic)}\end{bmatrix}_{3X3} \quad (1)$$
$$X \begin{bmatrix}\text{Transformation to align 3D}\\ \text{object coordinate systems to the}\\ \text{3D camera coordinate systems}\\ \text{(extrinsic transformation)}\end{bmatrix}_{3X4} X$$
$$\begin{bmatrix}\text{coordinates of the object point}\\ \text{(x, y, z)in 3D space}\\ \text{in homogeneous form}\end{bmatrix}_{4X1}$$

Image does not provide depth information and points on image contain only 2D information. In order to make equation (1) valid, we need to use homogeneous coordinates by assigning an arbitrary value (usually one) to z-coordinates in the location of the image plane. Homogeneous coordinates system is used in a wide range of applications, including computer graphics and 3D computer vision, in order to use matrix equation to model projective transformations correctly.

In a pinhole camera model, focal length f_x and f_y are very important parameters. Most cameras (including Ladybug3 cameras) have sensors with squared pixels, where $f_x = f_y$, therefore we only use f to represent focal length. In addition, image center (u_0, v_0) should also be taken into account.

Projecting a 3D point $\mathrm{P} = (X, Y, Z)$ on image plane at coordinates $\mathrm{P}' = (u, v)$ could be expressed by equation [19]:

$$\begin{cases} u = \dfrac{fX}{Z} + u_0 \\ v = \dfrac{fY}{Z} + v_0 \end{cases} \quad (2)$$

In the transformation of a point from its physical coordinates to the homogeneous coordinates, its dimension is augmented by introducing the scaling factor w [41]. Using homogeneous coordinates of P', equation (2) can be written as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3)$$

Transformation matrix in equation (3) corresponds to the 3 X 3 matrix in equation (1).

Usually, the origin of camera frame will not coincide with the origin in 3D world frame, and camera might be positioned at arbitrary angle. Therefore, rotation angle $(\theta_x, \theta_y, \theta_z)$ and translation (X_0, Y_0, Z_0) are also interesting parameters in final transformation equation. Rotation matrix could be represented as [30]

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (4)$$

$$= \begin{bmatrix} \cos\theta_z \cos\theta_y & \sin\theta_z \cos\theta_y & -\sin\theta_y \\ -\sin\theta_z \cos\theta_x + \cos\theta_z \sin\theta_y \sin\theta_x & \cos\theta_z \cos\theta_x + \sin\theta_z \sin\theta_y \sin\theta_x & \cos\theta_y \sin\theta_x \\ \sin\theta_z \sin\theta_x + \cos\theta_z \sin\theta_y \cos\theta_x & -\cos\theta_z \sin\theta_x + \sin\theta_z \sin\theta_y \cos\theta_x & \cos\theta_y \cos\theta_x \end{bmatrix}$$

, and translation is $\mathrm{T} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$.

Combining the two transformations above, a point in any 3D coordinate system which is either located at certain distance or oriented at an angle relative to camera coordinate system could be transform to points in 2D camera system through following equation

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \tag{5}$$

Using homogeneous coordinates for a 3D point P, equation (5) can be simply represented as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} [R \quad T] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Since Ladybug3 provides intrinsic parameters to users, matrix A which contains twelve unknowns has actually only 6 degree of freedom, namely rotation angles and translation. For a pair of correspondences $P(X, Y, Z)$ and $P'(u, v)$, they satisfy (a_1, a_2, a_3 denote row 1, 2, 3 of matrix A respectively) [30]

$$\begin{cases} \dfrac{u}{w} = \dfrac{a_1 P}{a_3 P} \\ \dfrac{v}{w} = \dfrac{a_2 P}{a_3 P} \end{cases} \tag{7}$$

Therefore, six pairs of such points would be enough to solve the 12 unknowns. However for better accuracy, much more than 6 correspondences are used in practice.

Transformation from local sensor system to Ladybug coordinate system should also be taken into account since Ladybug 3 has 6 sensors, each of which has its own local coordinate system. Matrix $M_i (i = 0,1,2,3,4,5)$ which is also a transformation matrix, indicating the transformation from coordinate system of sensor i to Ladybug head coordinate system. In the end, (6) will become

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = {M_i}^{-1} A \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{8}$$

Estimating six parameters $(\theta_x, \theta_y, \theta_z, X_0, Y_0, Z_0)$ for $[R \quad T]$ requires a non-linear minimization. Because of the complexity and risk of stopping at local minima in the six parameters estimation, we usually use more points to first estimate matrix A by solving the overdetermined homogeneous linear equation. It could be solved using singular value decomposition.

Non-linear optimization is usually required in this case. The most common method is use Levenberg–Marquardt algorithm to minimize projected position of a measured 3D position and the detected 2D position [8].

$$min_{R,T} = \frac{1}{2}\sum_{i=0}^{n} ||p_i - \hat{p}(R, T, P_i)||^2 \quad (9)$$

Where p_i is the detected 2D position of a point and $\hat{p}(R, T, P_i)$means the projected position of the point P_i.

This method has been applied in lots of previous work and it can usually produce good results. However, to solve the equation (5) together with the non-linear optimization is complicated and not so computationally efficient.

In fact, it is unnecessary to use the calibration model (5) since we do not need to estimate intrinsic parameters. Besides, multiple sensors of Ladybug and different local coordinate system of each sensor make the equation system (7) a bit complicated to build.

Preferably we can firstly transform image points from 2D coordinates to its Ladybug 3d ray (defined as its starting point and direction [17]) coordinates system and estimate pose of the checkerboard which is also known as PnP problem [39].

The output of the PnP solution are the depth factors of the camera points in the camera coordinate system. Then, we can recover the rigid transformation parameters between the two point sets through the correspondences within 3D space.

4.2.3 Pose estimation of checkerboard

Image does not contain any depth information. However it is possible to recognize 3-D objects from very few point features, and in fact, only one single view is sufficient to ascertain the pose of a 3-D object [29].

3D pose estimation is the problem of determining position and orientation of a 3D object in a 2D image. Position and orientation of objects with respect to the calibrated camera are the components of pose estimation. Given sufficient information in 3-D space and that in images, one can get the pose information of any object.

In our case, we want to get the pose of checkerboard. The 2D coordinates of all detected corner points on checkerboard are well known. Apparently extra information is required to estimate depth for projected points since depth information is lost due to the perspective projection. One can solve the problem by using the basic geometric constraints which relate distance $\mathrm{d}_{i,j}$ between points in the world reference frame together the scale factors λ_i, λ_j to the projected points P_i, P_j [33].

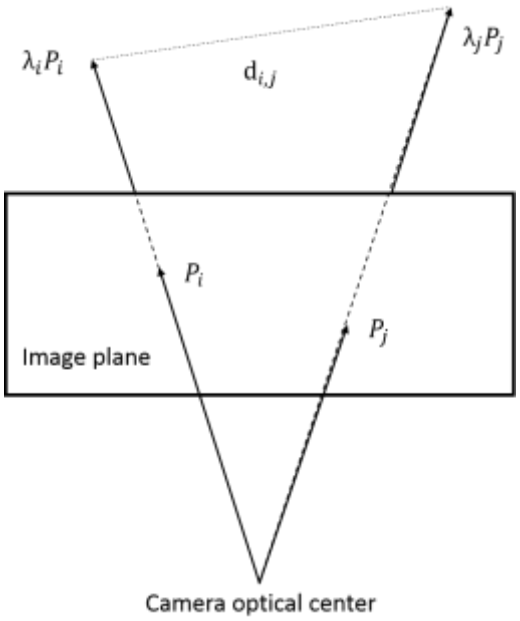

Figure 4-3: Pose estimation theory

Ladybug3 camera intrinsic parameters is provided so we could apply pinhole model to project a 2D point on a specific camera unit into a 3D ray in the Ladybug coordinates frame. The ray is emitted from its starting point along a certain direction.

A point in image 2D coordinate system is converted to local sensor's coordinate system simply by rewriting the equation (2) as

$$X = \frac{(u - u_0)}{f} \qquad (10)$$
$$Y = \frac{(v - v_0)}{f}$$

If we assume the image is projected on a plane in local coordinate system, any constant value could be assigned to Z, but usually we choose one. If one wants to project 2D points on a sphere, a constraint among $(X\_local, Y\_local, Z\_local)$ will be added

$$X_{local}^2 + Y_{local}^2 + Z_{local}^2 = R^2 \qquad (11)$$

R represents the radius of the sphere which an image is projected onto. It can be set to any positive value. 20 meter is the value we use because current Ladybug cameras are calibrated using a default sphere radius of 20 meters [17].

It does not matter which type of projection (planar, spherical, cylindrical…) we use since 3D rays emanated from the sensor are unchanged. The reason of using a spherical projection is panoramic image that we will use to extract color information is acquired using the spherical projection. Transforming points from plane onto a spherical surface is not a necessary step but just for the convenience to associate local coordinates to panoramic image plane.

One can get the value of Z in local camera coordinate system from the relation of similar triangles in figure:

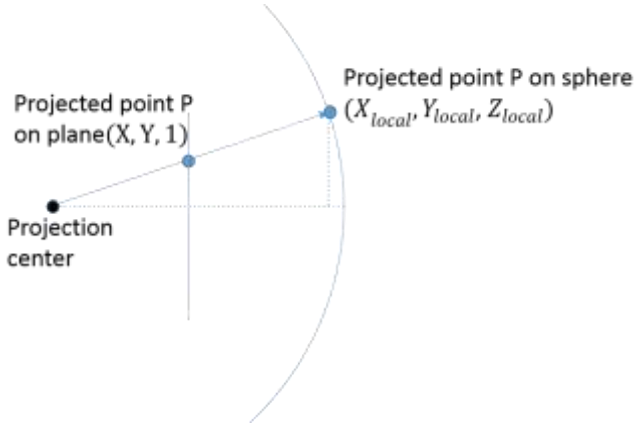

Figure 4-4: Computing coordinates on spherical surface

After we get the value of Z_{local}, X_{local} and Y_{local} will be acquired from equation (11). Point in 3D ray local coordinate system becomes

$$Z_{local} = \frac{R}{sqrt(X * X + Y * Y + 1.0)}; \quad (12)$$
$$X_{local} = X * Z_{local};$$
$$Y_{local} = Y * Z_{local}$$

Finally a transformation from local coordinate system to Ladybug coordinate system will be implemented with the provided transformation parameters.

In addition to the 3D ray coordinates of points in Ladybug camera coordinate system, an extra condition (e.g. measurements of objects in 3D world) should be offered for the depth estimation. Checkerboard will be the tool for depth estimation since all measurements of it are well known. We have all necessary conditions to estimate 3D coordinates of checkerboard,

- Known: Ladybug Ray Coordinates of point $\boldsymbol{Ray\_P_i}$ $(\boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots \boldsymbol{n})$on a sphere;

  Checkerboard square length l, width w;

  Number of square rows m, columns n

- Derived known: Distance d_{ij} between any two corner points$(\boldsymbol{Ray_P_i}, \boldsymbol{Ray_P_j})$

Next, equations could be built based on distance between pairs of points $(\boldsymbol{P_i}, \boldsymbol{P_j})$ on checkerboard.

- Equation:

$$|\lambda_i * \boldsymbol{Ray\_P_i} - \lambda_j * \boldsymbol{Ray\_P_j}| = d_{ij} \quad (13)$$

Any point on the board could be represented by its corresponding projected point scaled by a scale factor $\boldsymbol{P_i} = \lambda_i * \boldsymbol{Ray\_P_i}$

The unknowns wait to be solved are

- Unknown: Scale factor λ_n(n=1,2,3,4)

Three points are enough to constitute a plane. However, three points cannot generate unique solution. This is discussed as PnP ambiguity problem. Using only three point features usually lead to

four solutions [29] but in some special cases two solutions are possible. Figure 4-5 is a result when we use three points to build equations. Two solutions can meet the equations.

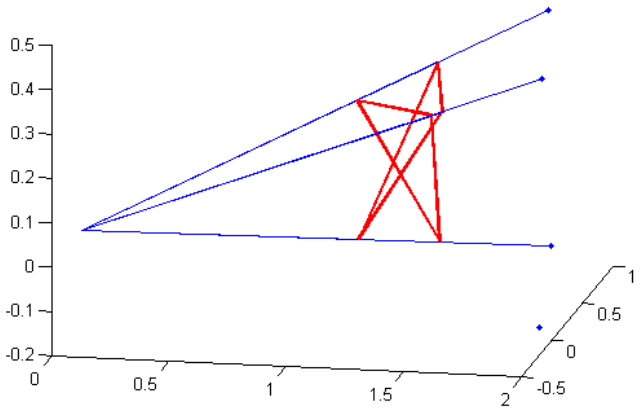

Figure 4-5: Using only three points might produce more than one result

Four coplanar points will give a unique solution in the case of perspective projection. If points are non-coplanar, six or more points should be employed in order to eliminate the ambiguity phenomenon.

In our case, points that lie on checkerboard are co-planar. Therefore at least four points are required. Four coplanar points can give six equations in total which will give a unique solution. In fact, according to our test, using four points can give sufficiently accurate result.

It is possible to directly obtain the unique solution from an overdetermined equation system in perspective four points problem [31]. However, due to small numerical error of the data, it does not apply in our case.

When more than three points are used, estimated points will not be exactly on the same plane if we simply using formula (13), for example, to build four equations with four points. That is because small errors exist in both automatic corner detection and coordinates transformation.

In order to make the estimated points on the same plane, we should using optimization algorithm to find an optimal solution. Either a Gradient Descent optimization algorithm or least square method can deal with this problem.

In this project, four points that can constitute a rectangle are chosen to build six equations (apply (13) to every two points)

$$\begin{cases} (\lambda_1 * \boldsymbol{P_1'} - \lambda_2 * \boldsymbol{P_2'})^2 = (n * w)^2 \\ (\lambda_3 * \boldsymbol{P_3'} - \lambda_4 * \boldsymbol{P_4'})^2 = (n * w)^2 \\ (\lambda_1 * \boldsymbol{P_1'} - \lambda_3 * \boldsymbol{P_3'})^2 = (m * l)^2 \\ (\lambda_2 * \boldsymbol{P_2'} - \lambda_4 * \boldsymbol{P_4'})^2 = (m * l)^2 \\ (\lambda_1 * \boldsymbol{P_1'} - \lambda_4 * \boldsymbol{P_4'})^2 = (n * w)^2 + (m * l)^2 \\ (\lambda_2 * \boldsymbol{P_2'} - \lambda_3 * \boldsymbol{P_3'})^2 = (n * w)^2 + (m * l)^2 \end{cases} \quad (14)$$

Next, we will search for the best solution for (14) by

1) Linearization of (14) with Taylor series, which is intended for approximating a function. Approximated function consists of a function $f(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ with its variables at certain value and derivatives with respect to each variable λ_i multiplied by infinitesimal Δ_{λ_i}

$$f(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = f(\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0) \quad (15)$$

$$+\frac{\partial f(\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0)}{\partial \lambda_1}\Delta_{\lambda_1} + \frac{\partial f(\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0)}{\partial \lambda_2}\Delta_{\lambda_1} + \frac{\partial f(\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0)}{\partial \lambda_3}\Delta_{\lambda_1} + \frac{\partial f(\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0)}{\partial \lambda_4}\Delta_{\lambda_1}$$

2) Give a set of initial values, e.g. $(0, 0, 1, 1)$ to $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$, use least square method iteratively untill$\Delta_{\lambda_i}$are small enough to be ignored.

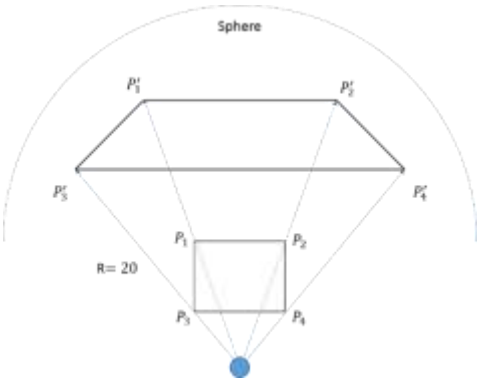

Figure 4-6: Depth estimation of checkerboard, front view

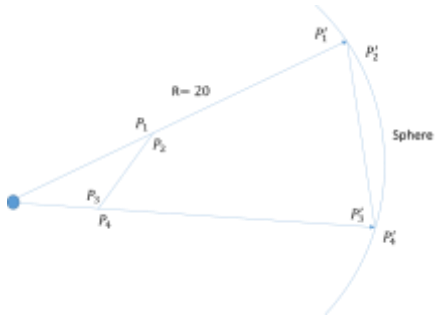

Figure 4-7: Depth estimation of checkerboard, side view

If multiple points are used, we will get a 3D plane model of the checkerboard through plane fitting.

4.2.4 3D rigid transformation

We are now ready to estimate parameters of the distortion-free transformation matrix (equation (4)), within the three-dimensional space. A comparison of several algorithms for computing 3-D rigid body transformations can be found in [34].

Assuming that a point $P_L\ (X_l, Y_l, Z_l)$ in laser coordinate system is transformed to a point $P_C\ (X_c, Y_c, Z_c)$ in camera coordinate system (points have the same unit, meter, in both Velodyne and Ladybug coordinate system), the relation is represented as

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} + T \quad (16)$$

R is a rotation matrix about rotation angle $(\theta_x, \theta_y, \theta_z)$ and T is the translation (X_0, Y_0, Z_0).

To determine rotation and translation in 3D space, three points must be known. In practice, more points would enable us to improve accuracy and to make a statistical assessment of the results and the calculation is adjusted with the least squares method, which firstly requires linearization of Equation (16).

$$P_C = T + R_L^C * P_L = T + \Delta R_L^C \tilde{R}_L^C P_L \quad (17)$$

After linearization, original matrix R_L^C (rotation from Laser coordinate system to camera coordinate system) is decomposed into an approximate matrix $\tilde{R}_L^C$ of main rotation and a matrix $\Delta R_L^C = \begin{bmatrix} 1 & \Delta r_z & -\Delta r_z \\ -\Delta r_z & 1 & \Delta r_z \\ \Delta r_z & -\Delta r_z & 1 \end{bmatrix}$ of small rotation. ΔR_L^C is derived from Equation (4) by applying approximate equations for small angles α: $\sin\alpha \approx \alpha$ and $\cos\alpha \approx 1$.

Approximate rotation matrix ΔR_L^C can be computed by using common points from both coordinate system [45]. First, choose three non collinear identical points from both coordinate system, and compute vectors between every two points in each system. Next, rotate vectors to compute angles and use quaternion to get rotation matrix. Quaternions [35] are used for represent rotation matrix in this case.

Finally, we can apply least square method to update R_L^C iteratively until small angles reduced to very trivial value.

We can simply use the geometric relation between vectors to get an initial value of rotation matrix without any complex computation compared to the method of getting initial transformation parameters by solving equation (7).

4.3 Evaluation method

We will use three methods to evaluate our results.

The first one is to measure the approximate value of the offset and rotation angles between two sensors. As for the rotation angles, we are not going to use measuring tools to measure them since it is difficult to implement. In fact, the two sensors are almost levelled since both of them stand on the floor. Meanwhile optical axis of Ladybug sensor 0 is nearly aligned with the Y axis of Velodyne laser scanner.

Therefore, rotation angle between two coordinates system (relative to camera system) will be close to $(0°, 0°, -90°)$.

Actually, when both sensors are levelled, two points would be enough to compute the transformation matrix. However, for more general uses, we still consider the case that all three rotation angles are unknown.

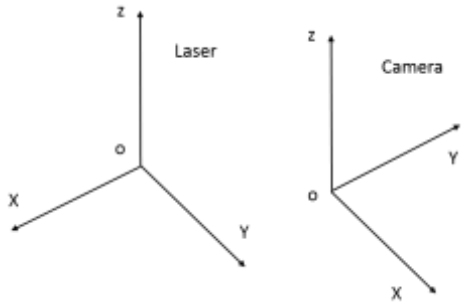

Figure 4-8: Coordinate system of laser scanner and camera

Several measurement are necessary for the computation of offset. Length need to be measured manually with a ruler. A Velodyne platform draft is available therefore we can get distance between any two points using a CAD software.

The following figures display all relevant measurements:

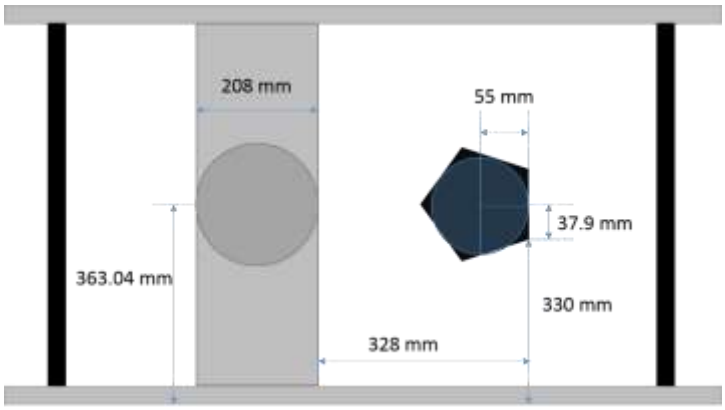

Figure 4-9: Sensors setup, top view

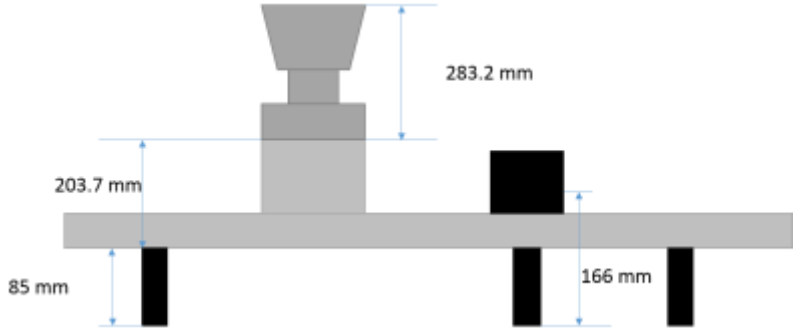

Figure 4-10: sensors setup, side view

An approximate offset relative to camera coordinate system (see Figure 4-9 and Figure 4-10) are calculates as follow:

X axis: $-328 + 55 - \frac{208}{2} = -377\ mm$

Y axis: $330 + 37.9 - 363.04 = 4.86\ mm$

Z axis: $203.7 + 85 - 166 = 122.7\ mm$

We will compare the calibration results with the ground truth in the end.

The second method is to check the projection error by computing the distance between projected points on image and their real position on the image.

Finally, we can check the result roughly through a visual inspection after we colorize the point cloud. Even though there is no convincing statistics are available in this approach, it is still a practical method to validate the result.

4.4 Data fusion

Using the transformation matrix obtained from (section), we first transform all points from Velodyne coordinate system to Ladybug3 coordinate system.

In order to find location for each point in panoramic image, we first convert Ladybug coordinate system from Cartesian coordinate system into spherical coordinate system.

Then a point P in the Cartesian coordinates (X, Y, Z) and the spherical coordinates (p, θ, φ), which represent radial distance, polar angle and azimuthal angle, are related as follows:

$$\begin{cases} p = \sqrt{X^2 + Y^2 + Z^2} \\ \theta = \arccos\left(\dfrac{Z}{\sqrt{X^2 + Y^2 + Z^2}}\right) \\ \varphi = arctan\dfrac{Y}{X} \end{cases} \tag{18}$$

If H and W represent height and width of a panoramic image respectively, the corresponding row r and column c of any point in the image can be computed from (p, θ, φ) using equation

$$\begin{aligned} r &= int\left[\frac{\theta}{\pi} * H\right] \\ c &= int[\left(\frac{\varphi}{2\pi} + 0.5\right) * W] \end{aligned} \tag{19}$$

Now all conditions that allow any point from point clouds being projected on a panoramic image are prepared. Next, we just project all points from point clouds to panoramic image. Color information (in the form of RGB value) will be extracted for all points at their location from the image and then will finally be added to point clouds.

4.5 Other Related Application

One of the task in this project is extracting the part of image corresponding to a given set of LIDAR points, which is also based on calibration work.

To extract the part of the image corresponding to the bounding box of the points projected onto the image plane, again, transformation matrix is required. This function is useful when we want to track or have a close inspection an object of interest.

A range of image is enclosed based on the range of projected object points. We simply take the combination of most left pixel and top pixel and the combination of most right pixel and bottom pixel as two vertices to create a rectangle to ensure all points within the interesting region are included.

Chapter 5 - Results

Final result of fused data and some essential middle results will be presented in this chapter, as well as evaluation and analysis of them.

Firstly, we will present a diagram of the whole program with input and output files and data flow to show all important procedure in this project and connections among them clearly.

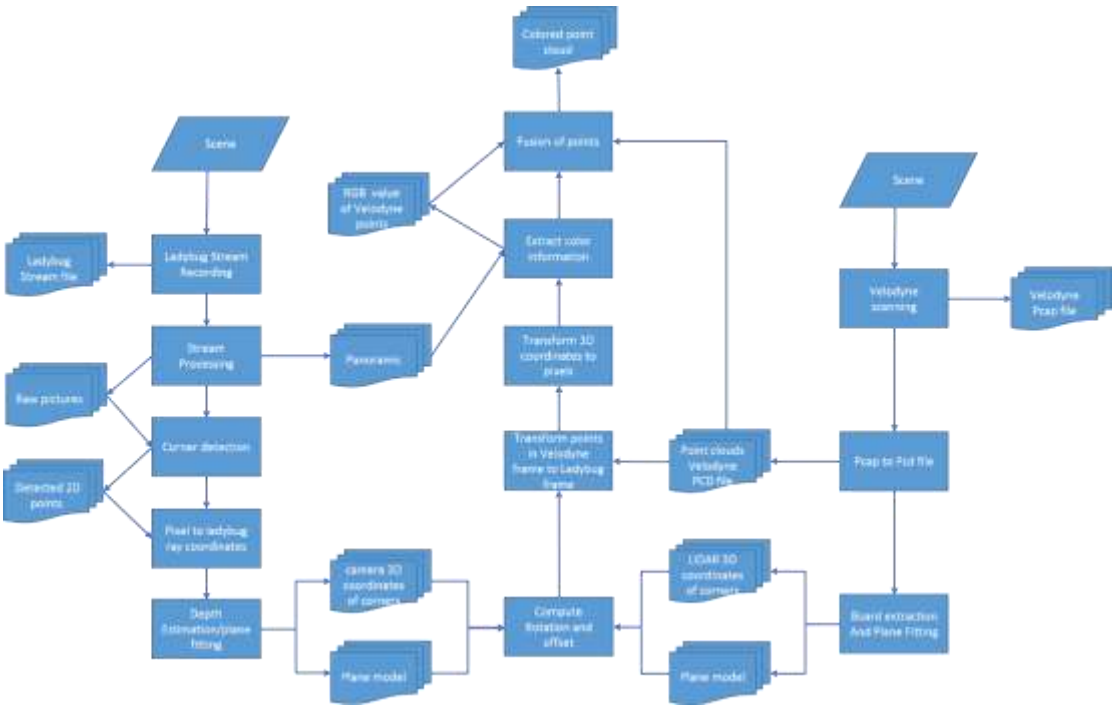

Figure 5-1: Program flow chart

A number of calibration related results will be displayed in this chapter. As for synchronization, we manage to generate timestamp logging files. However, because of our restricted experiment condition, we are unable to show any convincing results.

5.1 Calibration result

In this section, final calibration result (transformation parameters) and all intermediate result will be presented.

5.1.1 Board Extraction

The quality of the board extraction has big influence on calibration results. At it is mentioned in 4.2.3, all square corners will be computed based on the coordinates of board vertices. If segmentation of board gives bad result, then all points that we choose will have the same systematic error. In that case, using more points to compute transformation parameters will not really improve the result.

Figure 5-1 displays the extracted checkerboard and its border hull when it is settled at three different positions. Their corresponding images are shot by Ladybug sensor 0, sensor 1 and sensor 4 respectively (Figure 2-2).

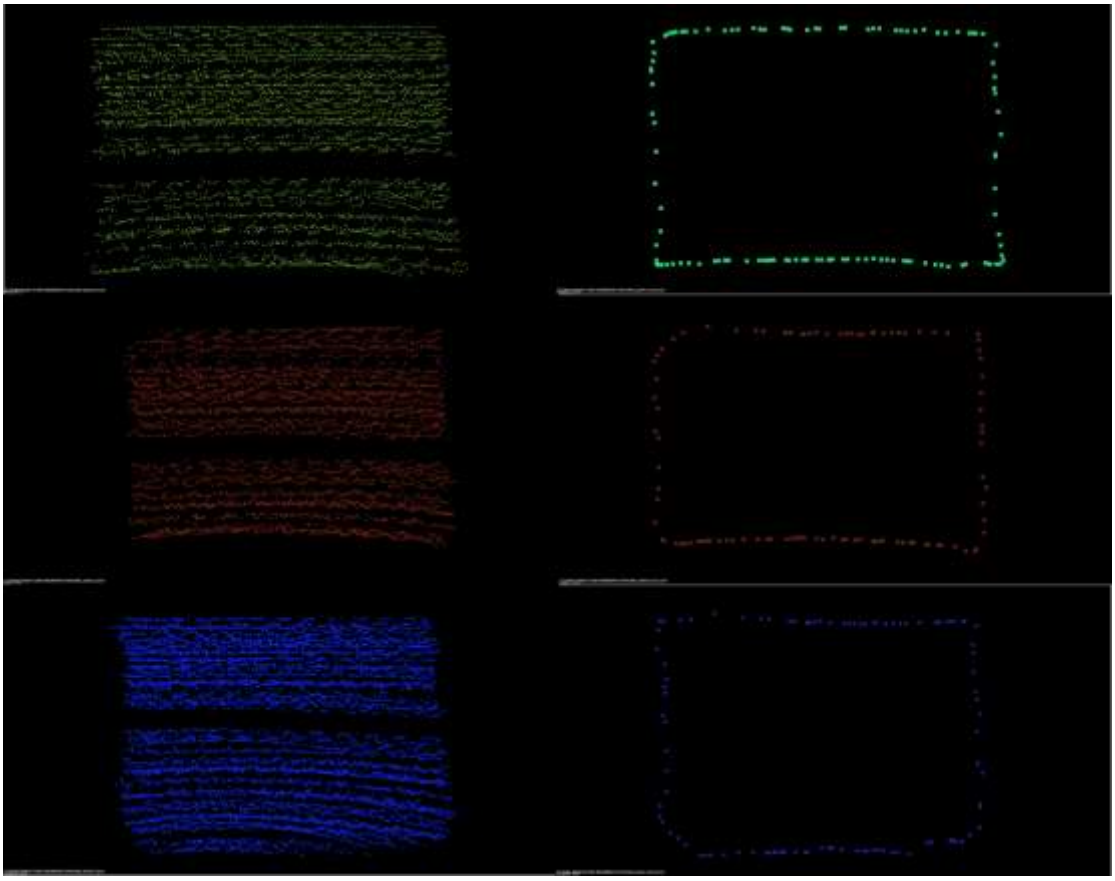

Figure 5-2: Board and its border, extracted at different positions

Through a visual check of Figure 5-2, the checkerboard is extracted correctly. However, it is not exact the original shape and this is particularly obvious in convex hull (right side of Figure 5-2). Just as what we explain in 3.2.2, noisy data makes feature extraction difficult. Therefore in our case, we are not able to give perfect input data for parameters estimation of transformation matrix.

Table 1 displays length of four border lines of the checkerboard extracted from point cloud. We name border lines from 1 to 4 as it is demonstrate in Figure 5-3.

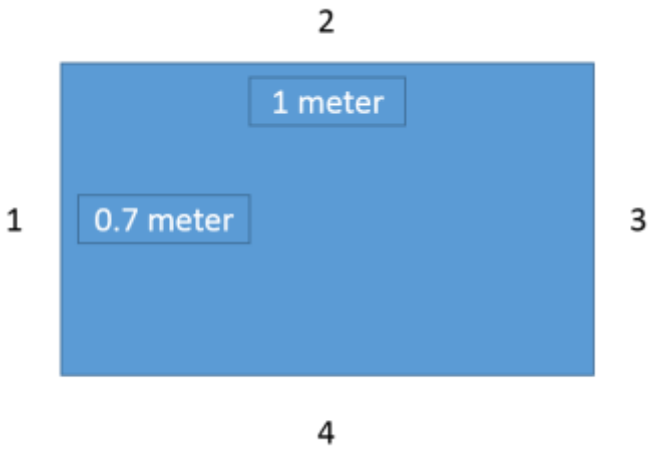

Figure 5-3: Order of board border lines

Border / Position	1	2	3	4
0	666.4 mm	1026.8 mm	717.0 mm	1020.5 mm
1	614.4 mm	1029.7 mm	685.1 mm	1013.4 mm
4	729.8 mm	1015.7 mm	747.6 mm	1051.4 mm

Table 1: Length of border lines of extracted board from point cloud

If we compare the value in table above with the real value (Figure 5-3), errors of the length of borders will be:

Border / Position	1	2	3	4
0	-33.6 mm	26.8 mm	17.0 mm	20.5 mm
1	-85.6 mm	29.7 mm	-14.9 mm	13.4 mm
4	29.8 mm	15.7 mm	47.6 mm	51.4 mm

Table 2: Error of border length of checkerboard

Errors are from one to eight centimeters in table above, and mostly around three centimeters. Detected border 2 and 4 tend to be longer than their real length and 1 is somehow always shorter than 3.

5.1.2 Corner detection of the board

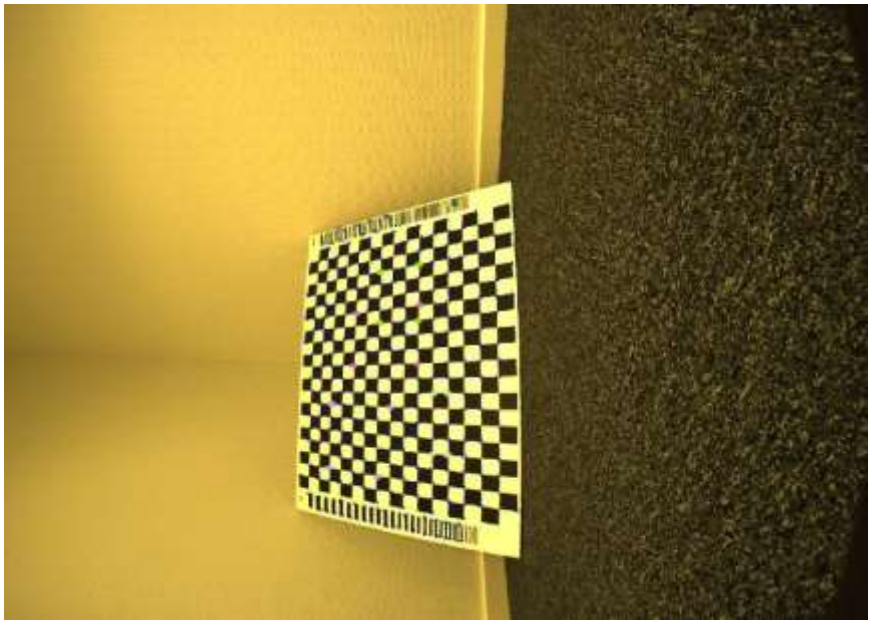

Figure 5-4: Corner detection

There is no significant result to show in the corner detection part. Small blue hollow circles are the corners detected automatically. Next, one can click on those detected points to choose some points for image depth estimation and computing transformation parameters. Chosen point will be marked with different colors to avoid confusion when the number of chosen points is big.

5.1.3 Depth estimation for the board

We choose four corner points which constitute a rectangle from image. Those points will be used to estimate the 3D coordinates of board in Ladybug coordinate system.

Below is a figure shows the estimated checkerboard at three different positions in camera coordinate system.

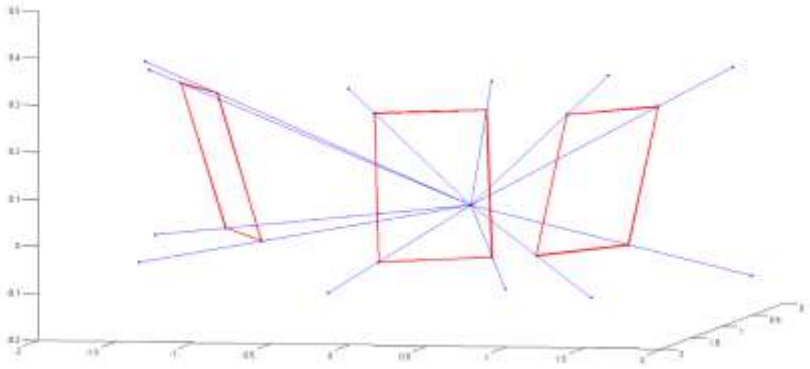

Figure 5-5: Estimated plane model for board from images

Figure 5-6 shows the real length and width of the rectangle whose vertices are selected corners. The green area of it corresponds to red rectangles in Figure 5-5.

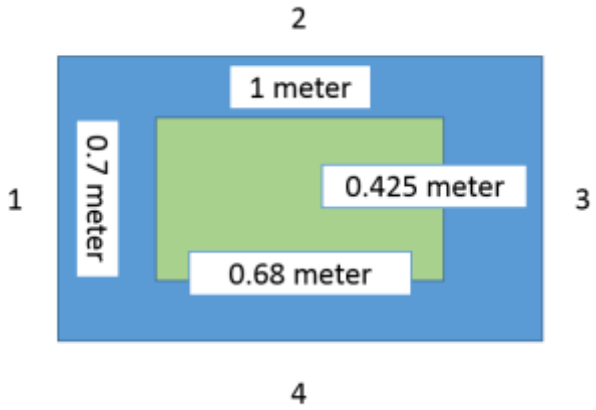

Figure 5-6: Checkerboard and the rectangle whose vertices are selected corners

Table 3 shows the estimated result for rectangle using least square method. Angle is a parameter to verify if all four estimated points lie on the same plane. This is computed from dot product between the normal of a rectangle plane (which is determined by any three vertices of the rectangle) and the vector from the fourth vertex of rectangle to any one of the other three vertices.

Border Position	1	2	3	4	Angle
0	428.9 mm	674.6 mm	426.8 mm	682.6 mm	90.03°
1	430.7 mm	671.3 mm	428.9 mm	684.0 mm	90.15°
4	422.8 mm	681.2 mm	425.8 mm	679.3 mm	89.87°

Table 3: Estimated length of every side of selected rectangle

Border Position	1	2	3	4	Angle
0	3.9 mm	-5.4 mm	1.8 mm	2.6 mm	0.03°
1	5.7 mm	-8.7 mm	3.9 mm	4. mm	0.15°
4	-2.2 mm	1.2 mm	0.8 mm	-0.7 mm	-0.13°

Table 4: Errors of estimated rectangle

Error of the estimated result are of millimeters order, from table 4, which mean it can achieve very good accuracy for board depth estimation.

5.1.4 Transformation Matrix

Recall the ground truth result we get from evaluation in 4.3.

Axis	Translation(millimeters)	Rotation angles(degree)
X	-377	0
Y	0.486	0
Z	122.7	-90

Table 5: Ground truth of rotation and offset (approximate value)

In this case, twelve points (same points as the ones used for depth estimation in 5.1.3) are chosen for computing transformation matrix.

Transformation Parameters					
Rotation Matrix			Axis	Translation	Rotation angles
-0.0279082	-1.00078	0.0038205	X	-399.37 mm	0.588698°
1.00863	-0.0345904	0.010381	Y	-35.74 mm	-0.218899°
-0.00982243	0.00526772	1.01031	Z	164.99 mm	-91.5974°

Table 6: Transformation parameters using points

Axis	Translation(mm)	Rotation angles
X	22.4	-0.5887°
Y	36.2	0.2189°
Z	-42.3	1.5974°

Table 7: Errors of transformation parameters (compared with ground truth)

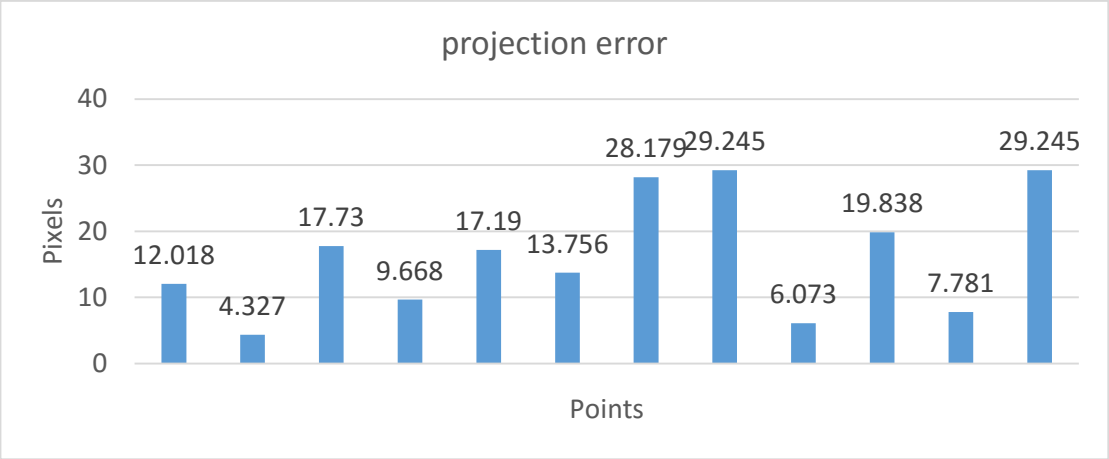

Figure 5-7: Projection error of some points

According to Table 7, the error of rotation around X, Y axis are within one degree and rotation around Z axis is 1.6 degree. This is an acceptable result considering that the 'ground truth' we measured is not accurate itself. Rotation around X, Y axis are almost close to zero since both of them are put on the same plane (floor). However the floor might not be completely flat itself which could cause small angles between two sensors along X and Y axis. Rotation around Z axis is very possible to be a bit away from zero degree because we place two sensors approximately perpendicularly by only visual check without any strict measuring. The translation errors is 2, 3, 4 centimeters along X, Y, Z direction, which is acceptable as well considering the errors happened in ground truth measurements and inaccuracy of board extraction.

Figure 5-7 is a projection error of several selected points using the computed transformation parameters. The largest error is up to 30 pixels. Considering the board extraction errors and the high resolution of our images, this result is reasonable.

We will apply this acceptable transformation parameters to display fusion result in the following sections.

5.2 Bounding box in image

In this section, we will just use an example to show the result from bounding box algorithm. Given a set of points from point clouds, we will find its corresponding points in image through registration if the transformation parameters acquired from calibration is correct. The bounding box which targets at the points set of interest will be marked and cropped.

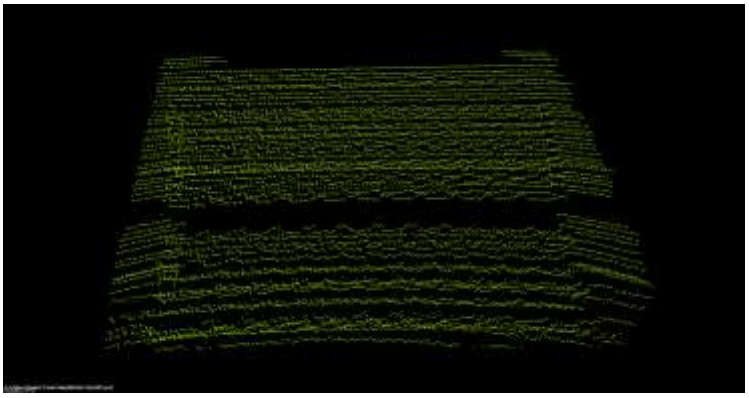

Figure 5-8: A given set of points (checkerboard)

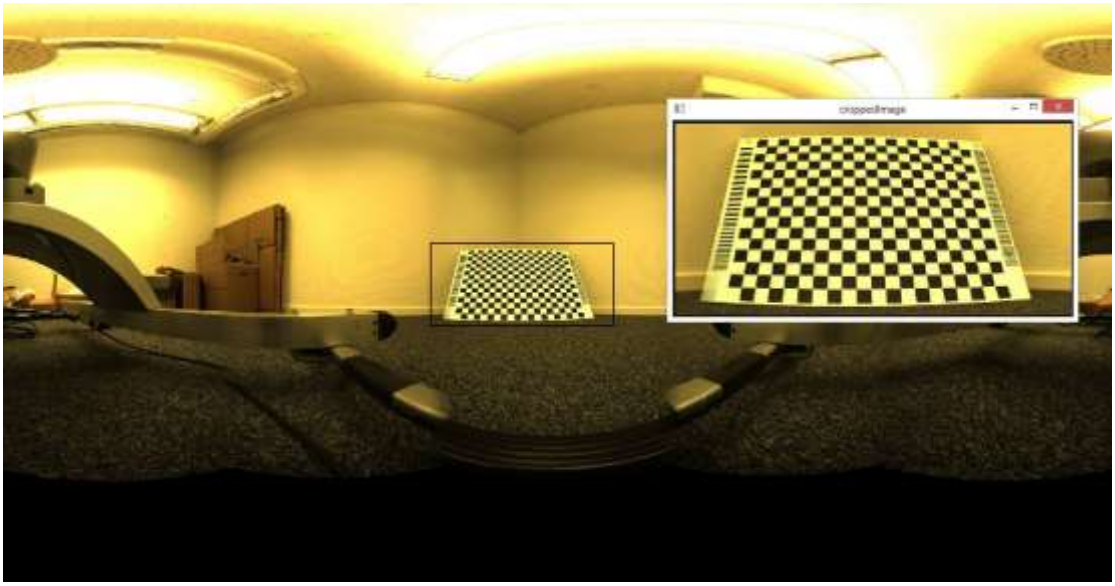

Figure 5-9: Bounding box of objects in image

5.3 Fusion of data

Fused data is the result of colored point clouds. Figure 5-10 to 5-18 presents scanned room with board at three different positions. Figure 5-19 is a point cloud which fused all three scans (Figure 5-10, Figure 5-14 and Figure 5-17).

It is difficult to detect objects other than board because the room is unorganized and the scanned points are very noisy. Unfortunately, the checkerboard is the only feature could be detected clearly and used to check the calibration result.

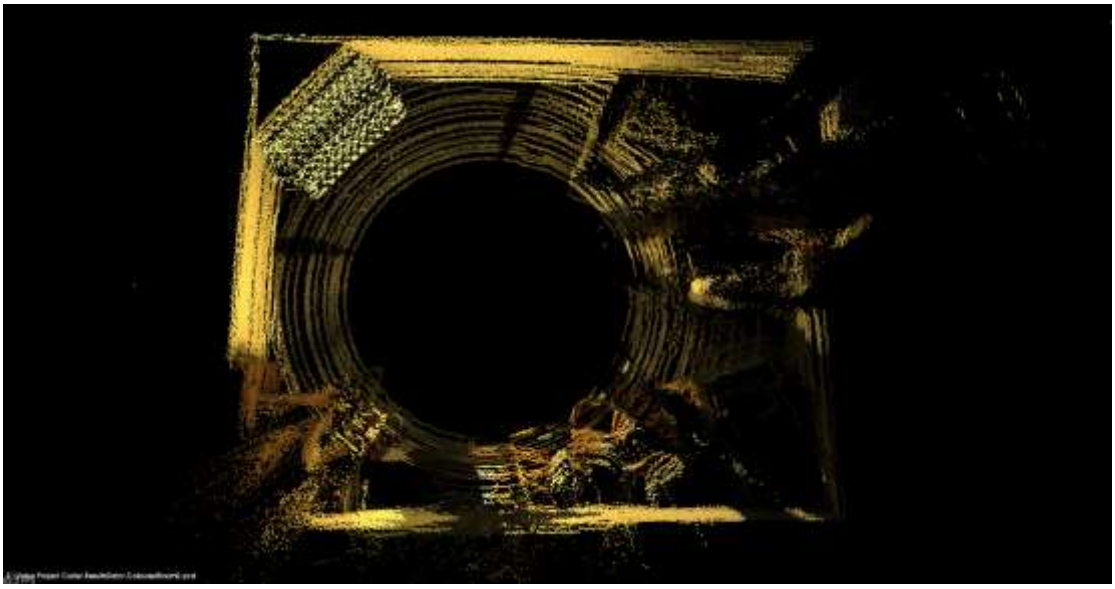

Figure 5-10: Scanned room where board at camera sensor 0 position

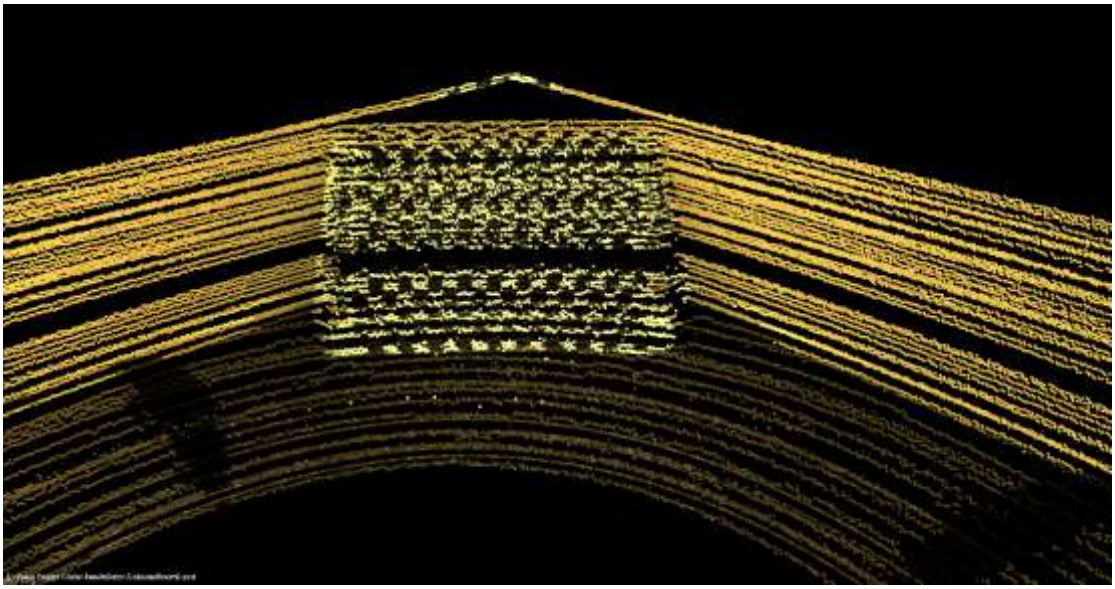

Figure 5-11: Board at camera sensor 0 position

Figure 5-10 is a close-up for the checkerboard. One can find that top edge of checkerboard does not get the checkerboard pattern (black and white) which is due to the small error of offset computation.

Meanwhile, it is interesting that the top edge of checkerboard does not get the checkerboard pattern (black and white) while some points even higher than checkerboard are colored with checkerboard pattern.

This could be explained by Figure 5-12. Laser scanner is set higher so it can view the blocked part of camera vision (see red line).

When the camera does a projection, depth information of points are lost. Therefore the points that are scanned by the laser scanner and are hidden behind the checkerboard from the view of camera will be projected on its corresponding 2D image plane as well and get the color information from that position.

Figure 5-13 shows approximately the way board is viewed by sensor 0, and that corresponds to the explanation in Figure 5-12.

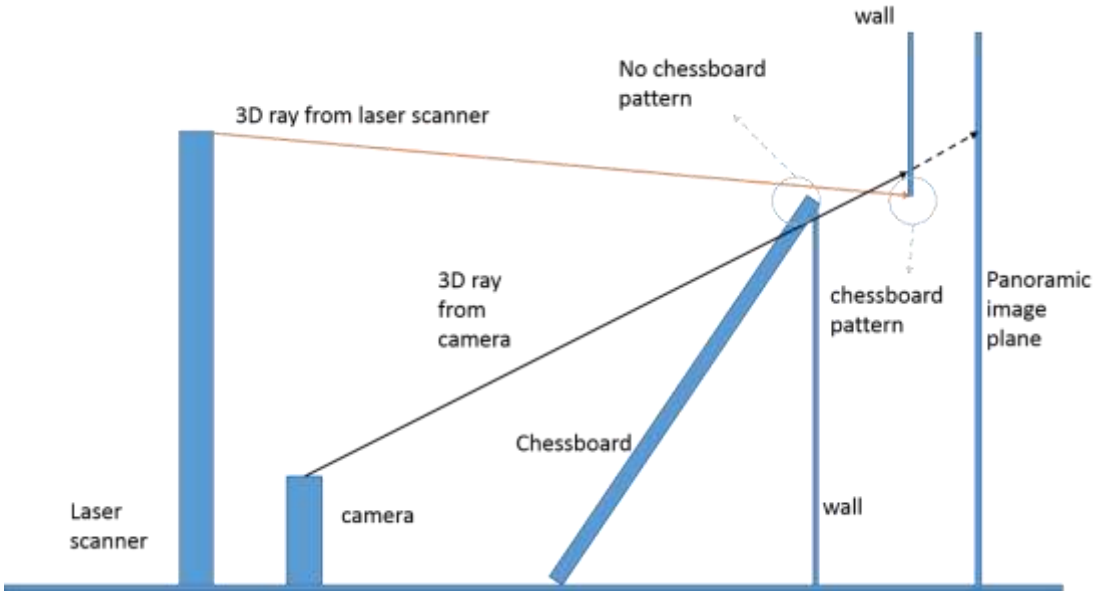

Figure 5-12: Explanation of projection problem

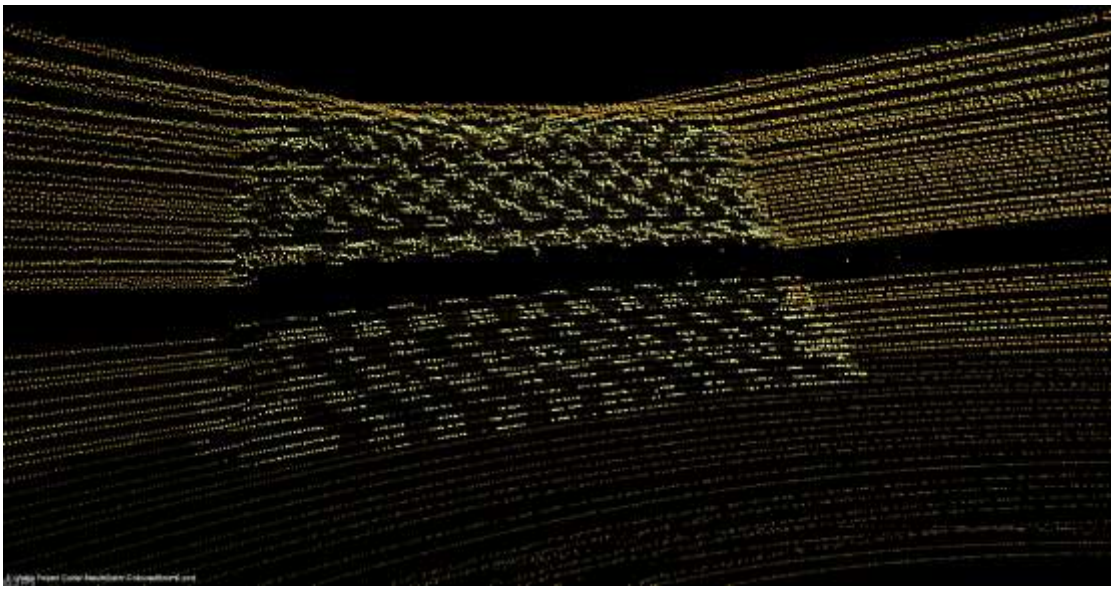

Figure 5-13: Board at camera sensor 0 position

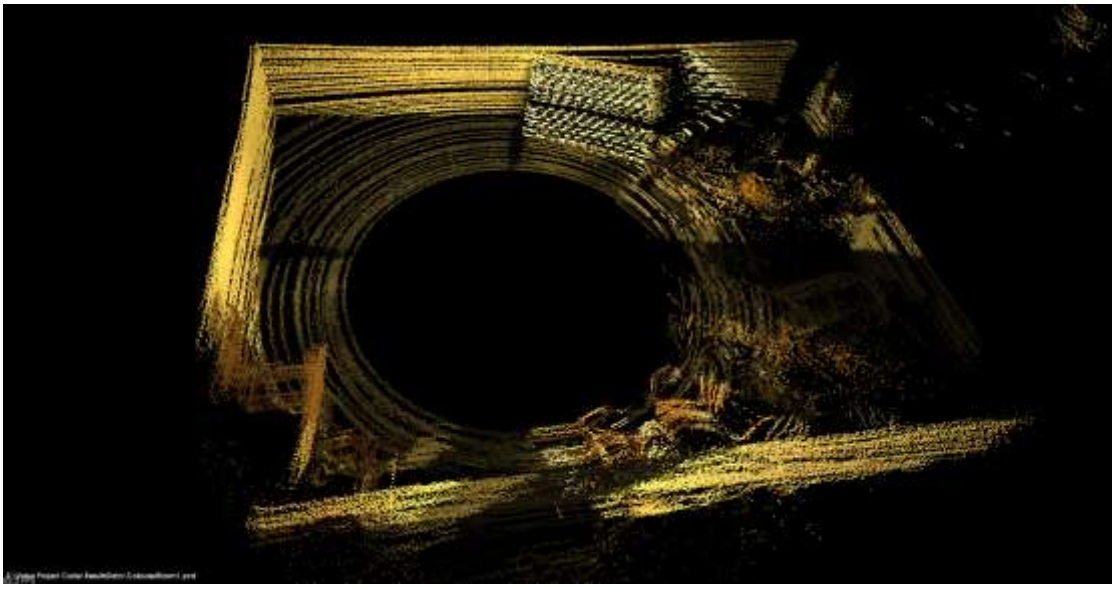

Figure 5-14: Scanned room where board at camera sensor 1 position

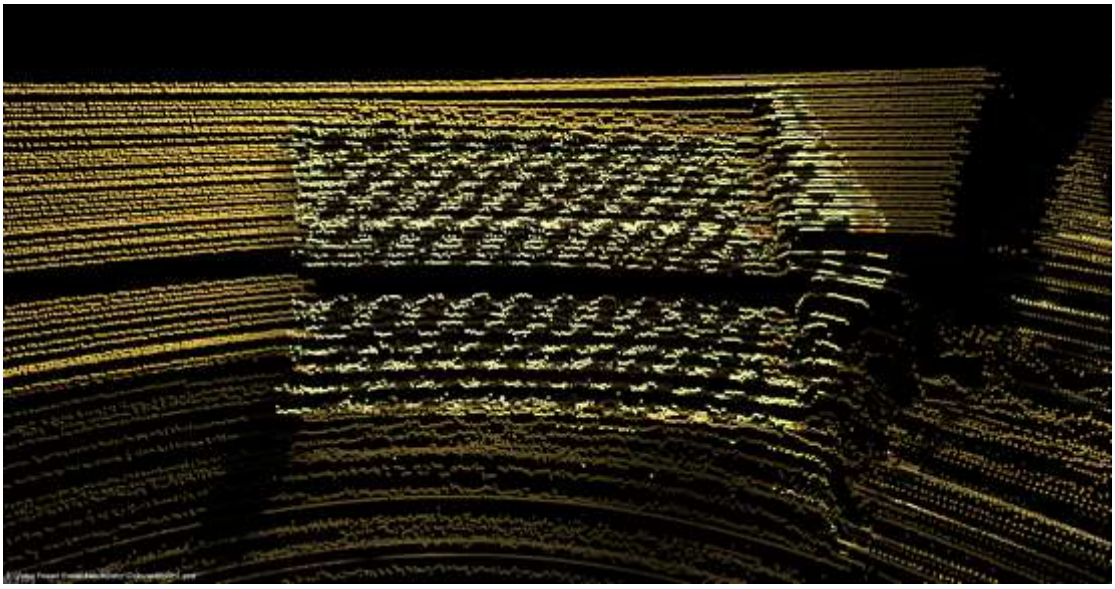

Figure 5-15: Board at camera sensor 1 position

In Figure 5-15, one can find that the right edge of board missed checkerboard color which is due to errors in transformation parameters. Again some part out of the checkerboard plane get the black and white pattern. This could be explained by Figure 5-12 as well, but on the horizontal direction instead of vertical direction in Figure 5-11.

Figure 5-16 shows approximately the way board is viewed by sensor 1, which corresponds to the explanation in Figure 5-12.

Figure 5-16: Board at camera sensor 1 position

In Figure 5-17 and 5-18, one can see that bottom of board has an opening without checkerboard pattern. This is because a beam of platform lie in the front of sensor 4 (check Figure A-3 and Figure A-6 in appendix A).

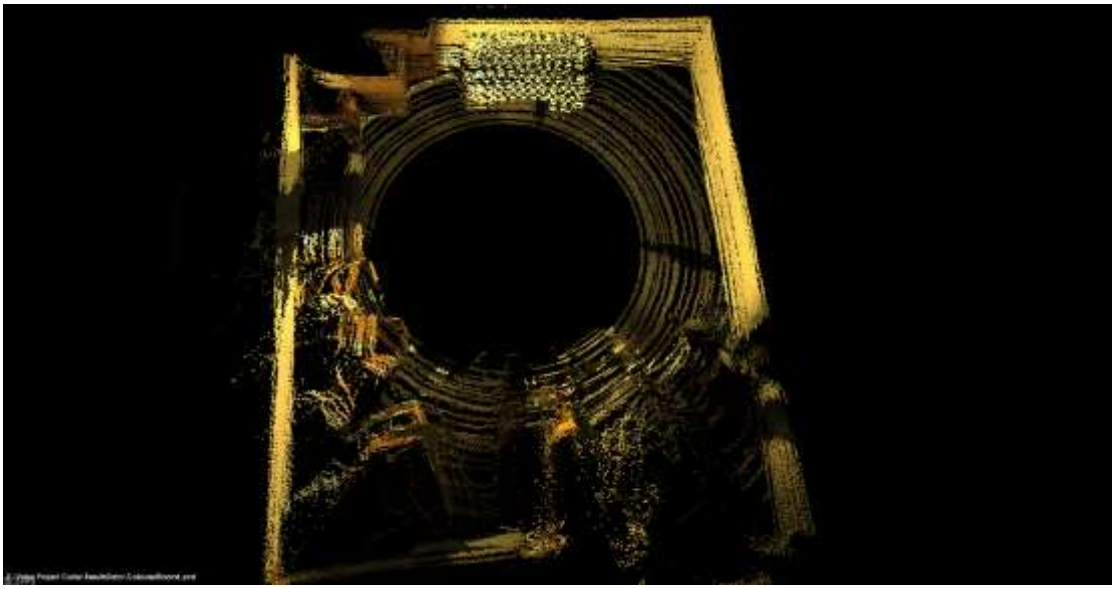

Figure 5-17: Scanned room where board at camera sensor 4 position

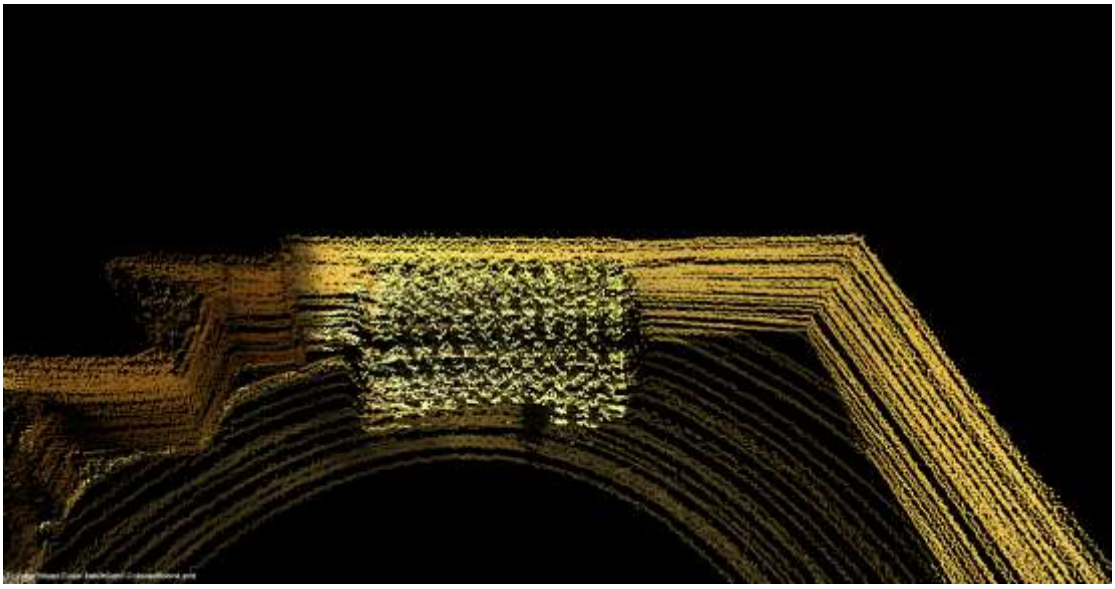

Figure 5-18: Board at camera sensor 4 position

Finally, a fused point cloud made of all above three scans presented in Figure 5-19. One can compares this with Figure 5-5, and panoramic images in appendix A. The mapping results looks reasonable.

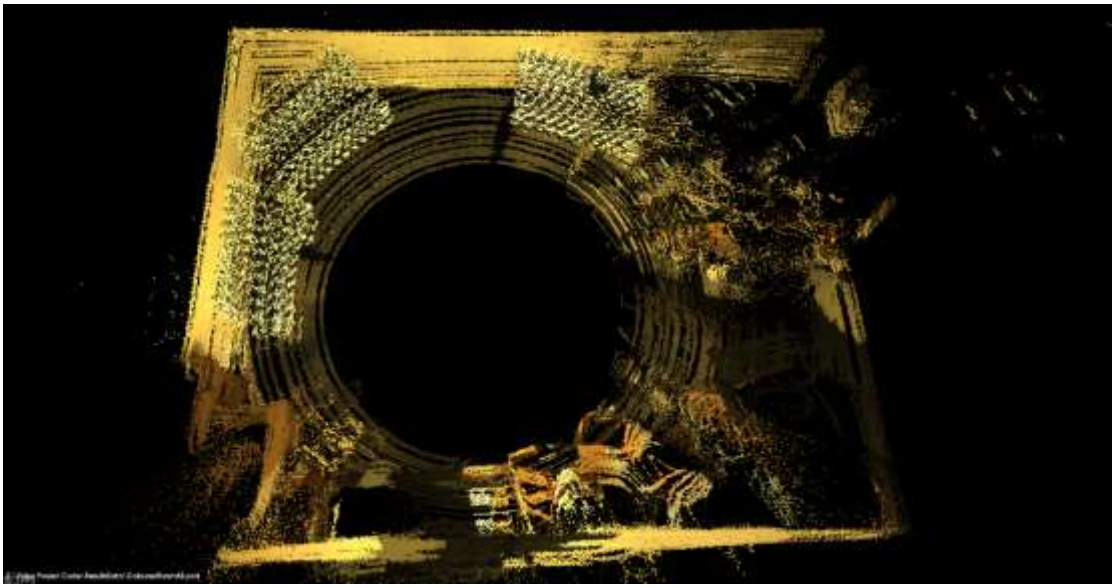

Figure 5-19: Fusing all scans together

Chapter 6 - Discussion

In this chapter, the problems of the calibration work will be analyzed based on the results we get. Applied calibration method will also be commented. In the end, results of similar work accomplished by others will be compared with ours.

6.1 Data quality

The mapping result is not so accurate even though the transformation parameters are approximately correct compared with ground truth. There are few factors which might reduce the accuracy and reliability of the data fusion results -

Firstly, the laser points are noisy because they are not perfectly calibrated which makes the board extraction inaccurate. Board extraction algorithm plays a role as well. In our project, we are unfortunately unable to manage trying out all possible segmentation method because of the time limit. According to the board extraction result, one can easily find that this is the main error source. It is possible to improve the feature extraction result with an alternative algorithm.

Furthermore, we calibrate the sensors without applying refining algorithm such as iterative closest point method. In fact, it does not really help a lot if the target point clouds fail to offer accurate information itself, which is the checkerboard in this case. However, refining algorithm will improve the calibration result if not considering the quality of data input.

The inconsistent angle of view of laser scanner and camera as it is addressed in Figure 5-12 might cause image 'see through' objects that is blocked. This will cause mismatch of color and the real objects. For example, the chessboard pattern shows up in peripheral area of chessboard Figure 5-15. In order to solve this we need to remove those redundant points that are blocked from the view of camera beforehand.

6.2 Method

We employ a calibration objects and use 3D Points reference to calibrate camera and laser scanner extrinsically. Even though other methods which are less demanding on the calibration apparatus exists, more complicated mathematical computation should be done in those cases. Furthermore, using 3D geometry of objects can generate the most accurate result. Since we have an existing checkerboard, using 3D point reference to implement calibration is the best choice in our experiment.

The method of computing the rigid transformation parameters between camera and laser scanner is a little different from the geometric camera calibration method usually used in other articles. Instead of applying geometric calibration model and solving equation of matrices by taking advantage of the projection relation in camera model, rotational components of the transformation parameters are firstly computed using three pairs of known correspondences, followed by a linear optimization using least square formulation to get more accurate rotation angle and compute translation. In this method, we need to know the 3D coordinates of points in camera reference frame and that is why a pose estimation of checkerboard is done before computing 3D rigid parameters.

A general solution is to first use one direct linear solution called Direct Linear Transformation (DLT) then apply a non-linear optimization algorithm. DLT formulates a homogeneous linear system of equations and solution of the system is usually found by eigenvalue computations using technique known as constrained optimization.

The way to solving DLT is not physically meaningful, therefore a non-linear methods for estimating follows the DLT. It will cost some time in this computation process.

In addition, Equation (7) may lead to very ill-conditioned matrix equation which might cause problem when finding its solution. Therefore, compared to the method of applying geometric camera calibration model, our method is computationally easier, faster and there is no need to worry about invalid solution which might exist when solving equation of camera calibration model.

The inaccurate input data for computing transformation parameters make it difficult to evaluate the method itself. We first need Figure out a way to get accurate points correspondence in order to avoid the interference from data. Further processing work will continue.

6.3 Result of other papers

Experiments in paper [4] show that the method can project the laser points onto the image plane with an average error of 1.01 pixels (1.51 pixels worst case). The biggest error in our result is up to 30 pixels. However, the camera they use has a horizontal resolution of 640 pixels which is not comparable with ours 1600 X 1200 pixels. In their experiment, well-spaced target are employed. In contrast, we use much smaller target points (corners of checkerboard) concentrated in a place. All corners on checkerboard in point cloud have systematic error because of the imperfect board extraction result.

A similar method of calibrating Velodyne and a camera is presented in paper [6]. When the number of views is three, the projection error varies from four to twenty and their color camera with resolution of 659 × 493.

In paper [9], the result displays error in rotation is 0.6 degree and in translation is 0.04 m when number of view is three (since we only use three views in the experiment), which is close to the result we get.

The experiment setup for acquisition of data and the quality of the acquired data are different in different papers. It is not a fair comparison among the papers by simply looking at the results. In our project, we implement an experiment with a very simple setup and collect very limited amount of data which can be put into use. Besides, we are not confident about the accuracy of data acquired from Velodyne Lidar.

However, for the purpose of validating the method we use, the result can be a support for our method since we manage mapping color onto point cloud at correct position if considering the overall performance and errors are within acceptable range provided that the error in board extraction.

Chapter 7 - Conclusion

Synchronization and calibration of sensors are the two most important steps of fusing camera image and point clouds, which are also the two essential work performed in this project. Firstly, a hardware platform is built for data collection. It takes a bit effort to build connection between GPS receiver and laptop because we need to wire a connection line ourselves.

Synchronization in this project is to find data acquired from different sensors at the same time by matching the timestamps. Since this work could be done offline, it does not require synchronization of sensors in real time. Therefore, the timestamps will be inserted into different datasets when the sensors record data. After that, a series of steam processing work will allow accessing the timestamp.

Once we match datasets correctly, a calibration between Ladybug3 camera and Velodyne laser scanner is performed using the data of exact the same environment. Our calibration method mainly focuses on extrinsic calibration between camera and laser scanner. The intrinsic calibration is left out by taking advantage of the intrinsic parameters offered by manufacturer. Instead of applying 3D to2D projection model and solve equation to get unknowns, we will compute transformation parameters within 3D space and then use optimization algorithm to improve the computed transformation parameters.

Finally, the images are mapped into their corresponding point clouds, producing a relatively good result in comparison with that in other literature.

In the future, we might continue our research from the following aspects:

1) Collecting data in field and mapping image into point clouds with the calibration and synchronization result we get.
2) Improving present method using other possible optimization algorithms
3) Adding texture to point clouds.
4) Fuse texture-data from different distances to create *mipmap* textures
5) Estimate BRDF (bidirectional reflectance distribution function)-material properties for surfaces and different objects e.g. car is shiny road is matte, lane markings have high specular properties.

This project offers a method of fusing different types of data in dynamic environment, which is also the fundamental work for improving the perception of autonomous vehicle. The result of our method turns out to be viable for further applications even though it is currently not impeccable due to problems of sensors which may take a bit extra work to solve and some operational mistakes in experiment setup which are easily avoided next time. Algorithm of calibration also contributes to the quality of final result, but in this project, its influence is trivial compared to that of the quality of point correspondences. In the future work, we will both improve the data quality and meanwhile optimize the algorithm to make the result as accurate as possible.

Bibliography

[1] Koschorrek, P.; Piccini, T.; Oberg, P.; Felsberg, M.; Nielsen, L.; Mester, R., "A Multi-sensor Traffic Scene Dataset with Omnidirectional Video," Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on , vol., no., pp.727,734, 23-28 June 2013

[2] Synchronised Data Acquisition for Sensor Data Fusion in Airborne Surveying

[3] Langley, Richard. "NMEA 0183: A GPS Receiver." GPS world (1995).

[4] Osgood, Thomas J., and Yingping Huang. "Calibration of laser scanner and camera fusion system for intelligent vehicles using Nelder–Mead optimization." Measurement Science and Technology 24.3 (2013): 035101.

[5] Glennie, C.; Lichti, D.D. Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning. Remote Sens. 2010, 2, 1610-1624.

[6] Park, Y.; Yun, S.; Won, C.S.; Cho, K.; Um, K.; Sim, S. Calibration between Color Camera and 3D LIDAR Instruments with a Polygonal Planar Board. Sensors 2014, 14, 5333-5353.

[7] Mostafa, Mohamed MR. "Boresight calibration of integrated inertial/camera systems." Proc. International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation–KIS. 2001.

[8] Pless, Robert, and Qilong Zhang. "Extrinsic calibration of a camera and laser range finder." IEEE International Conference on Intelligent Robots and Systems (IROS). Vol. 13. 2004.

[9] Pandey, Gaurav, et al. "Extrinsic calibration of a 3d laser scanner and an omnidirectional camera." *7th IFAC Symposium on Intelligent Autonomous Vehicles*. Vol. 7. No. 1. 2010.

[10] Mirzaei, Faraz M., Dimitrios G. Kottas, and Stergios I. Roumeliotis. "3D LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization." The International Journal of Robotics Research 31.4 (2012): 452-467.

[11] Pandey, Gaurav, et al. "Automatic Targetless Extrinsic Calibration of a 3D Lidar and Camera by Maximizing Mutual Information." AAAI. 2012.

[12] Zhang, Zhengyou. "A flexible new technique for camera calibration." Pattern Analysis and Machine Intelligence, IEEE Transactions on 22.11 (2000): 1330-1334.

[13] Wang, Ruisheng, Frank P. Ferrie, and Jane Macfarlane. "Automatic registration of mobile LiDAR and spherical panoramas." Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on. IEEE, 2012.

[14] Scaramuzza, Davide, Ahad Harati, and Roland Siegwart. "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes." Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE, 2007.

[15] Unnikrishnan, Ranjith, and Martial Hebert. "Fast extrinsic calibration of a laser rangefinder to a camera." (2005).

[16] “Ladybug3™ Technical Reference Manual.” Point Grey. Point Grey Research. 8 March, 2011. Web. 26 June 2015. <http://www.ptgrey.com/support/downloads/10127/>

[17] “Geometric Vision using Ladybug Cameras.” Point Grey. Point Grey Research. 20 January, 2014. Web. 26 June 2015. <https://www.ptgrey.com/tan/10621>

[18] “HDL-64E.” Velodyne. Velodyne Lidar, 2013. Web. 26 June 2015. <http://velodynelidar.com/lidar//hdlproducts/hdl64e.aspx>.

[19] “Camera pinhole model”. Wikipedia. Wikimedia Foundation, Inc, April 1, 2015. Web. 20 May, 2015. <http://en.wikipedia.org/wiki/Pinhole_camera_model>

[20] Kocmanova, Petra, Ludek Zalud, and Adam Chromy. "3D proximity laser scanner calibration." Methods and Models in Automation and Robotics (MMAR), 2013 18th International Conference on. IEEE, 2013.

[21] “Module sample_consensus.” Point Cloud Library (PCL). 15 Dec, 2013. Web. 26 June 2015. <http://docs.pointclouds.org/1.7.0/group__sample__consensus.html>.

[22] Mei, C.; Rives, Patrick, "Calibration between a central catadioptric camera and a laser range finder for robotic applications," Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on , vol., no., pp.532,537, 15-19 May 2006

[23] The KITTI Vision Benchmark Suite. Karlsruhe Institute of Technology. Web. May 20, 2015. <http://www.cvlibs.net/datasets/kitti/raw_data.php>

[24] Ikeda, Sei, Tomokazu Sato, and Naokazu Yokoya. "Calibration method for an omnidirectional multicamera system." Electronic Imaging 2003. International Society for Optics and Photonics, 2003.

[25] Geiger, A.; Moosmann, F.; Car, O.; Schuster, B., "Automatic camera and range sensor calibration using a single shot," Robotics and Automation (ICRA), 2012 IEEE International Conference on , vol., no., pp.3936,3943, 14-18 May 2012

[26] Shi, J.; Tomasi, C., "Good features to track," Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on , vol., no., pp.593,600, 21-23 Jun 1994

[27] Li, Hao, and Fawzi Nashashibi. "Comprehensive extrinsic calibration of a camera and a 2D laser scanner for a ground vehicle." (2013): 24.

[28] Vasconcelos, F.; Barreto, J.P.; Nunes, U., "A Minimal Solution for the Extrinsic Calibration of a Camera and a Laser-Rangefinder," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.34, no.11, pp.2097,2107, Nov. 2012

[29] E.R. Davies, Chapter 16 - Tackling the Perspective n-point Problem, In Computer and Machine Vision (Fourth Edition), edited by E.R. Davies, Academic Press, Boston, 2012, Pages 424-438, ISBN 9780123869081, <http://dx.doi.org/10.1016/B978-0-12-386908-1.00016-1>.

[30] E.R. Davies, Chapter 18 - Image Transformations and Camera Calibration, In Computer and Machine Vision (Fourth Edition), edited by E.R. Davies, Academic Press, Boston, 2012, Pages 478-504, ISBN 9780123869081, <http://dx.doi.org/10.1016/B978-0-12-386908-1.00018-5>.

[31] Quan, Long, and Zhongdan Lan. "Linear n-point camera pose determination." Pattern Analysis and Machine Intelligence, IEEE Transactions on 21.8 (1999): 774-780.

[32] Zheng, Yinqiang, et al. "Revisiting the pnp problem: A fast, general and optimal solution." *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013.

[33] Ansar, Adnan, and Kostas Daniilidis. "Linear pose estimation from points or lines." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.5 (2003): 578-589.

[34] Eggert, David W., Adele Lorusso, and Robert B. Fisher. "Estimating 3-D rigid body transformations: a comparison of four major algorithms." *Machine Vision and Applications* 9.5-6 (1997): 272-290.

[35] Horn, Berthold KP. "Closed-form solution of absolute orientation using unit quaternions." *JOSA A* 4.4 (1987): 629-642.

[36] Oxford Technical Solutions. "RT3000 User Manual." ifor. Intelligent Off-Road Vehicles, 28 July, 2003. Web. 26 June, 2015. <http://www8.cs.umu.se/research/ifor/dl/Product%20info/OTS%20RS3000/rt3kman.pdf >.

[37] Garmin International, Inc. "GPS Technical specifications 18X." Garmin. Garmin Corporation. October, 2011. Web. 26 June, 2015. <http://static.garmincdn.com/pumac/GPS_18x_Tech_Specs.pdf>.

[38] "USER'S MANUAL AND PROGRAMMING GUIDE." Velodyne. Velodyne Lidar, 2013. Web. 26 June 2015. <http://velodynelidar.com/lidar/products/manual/HDL-64E%20S3%20manual.pdf>.

[39] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge Univ. Press, 2nd edition, 2003.

[40] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[41] Sinha, Pradip K. (2012). Image Acquisition and Preprocessing for Machine Vision Systems. SPIE. Online version available at: http://app.knovel.com/hotlink/toc/id:kpIAPMVS03/image-acquisition-preprocessing/image-acquisition-preprocessing

[42] Zhang, Z. (2005), Camera calibration, in Emerging Topics in Computer Vision, edited by G. Medioni, and S. B. Kang, pp. 4–43, Prentice Hall, Upper Saddle River, N. J.

[43] Harry Lum & Jerry A. Reagan (Winter 1995). "Interactive Highway Safety Design Model: Accident Predictive Module". Public Roads Magazine.

[44] Marc Weber. "Where to? A History of Autonomous Vehicles" Computer History Museum. Web. 30 Mar. 2015. < http://www.computerhistory.org/atchm/where-to-a-history-of-autonomous-vehicles/>.

[45] Milan Horemuz. "Determination of transformation parameters." Course: Laser scanning. Royal Institute of Technology Institute of Infrastructure, Division of Geodesy. September 5, 2014. Web. April 15, 2015.

Appendix A- Point clouds and Images

Extracted points clouds and images that we use in this project are listed here.

A.1 Raw images and panoramic images

Figure A-1: Raw image of board shot by camera 0

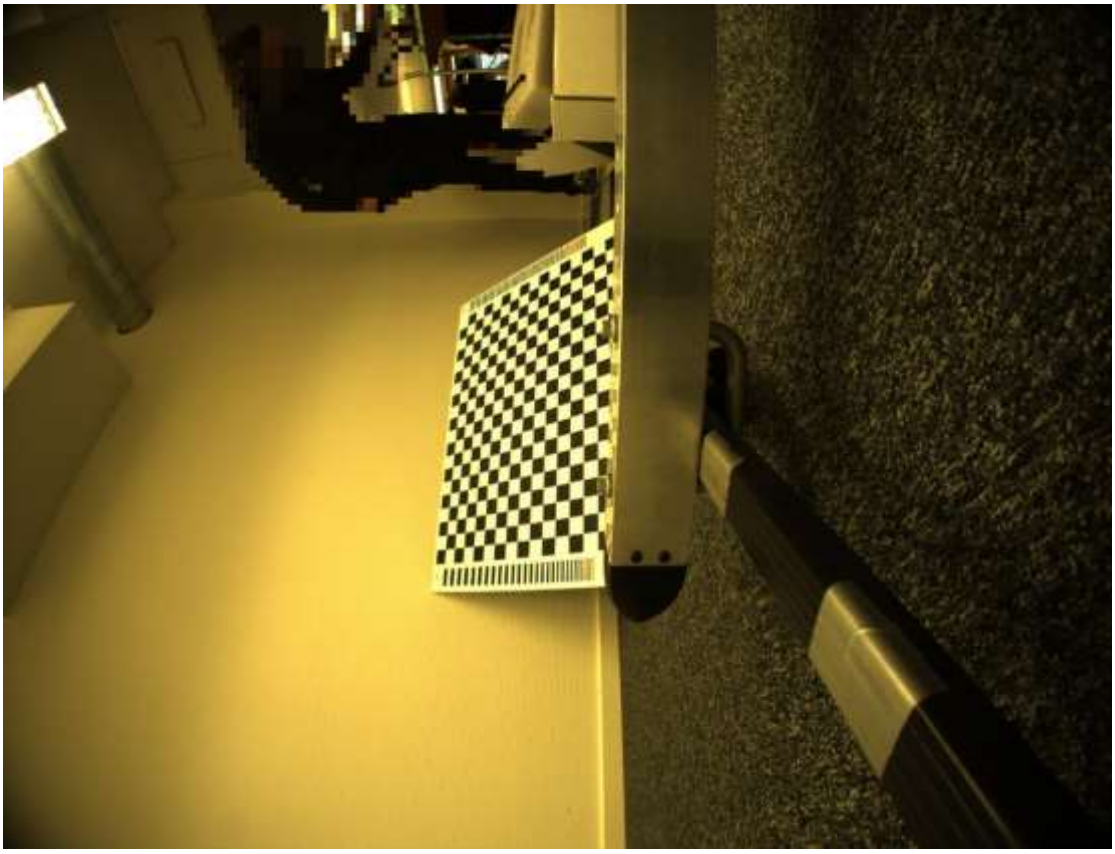

Figure A-2: Raw image of board shot by camera 1

Figure A-3: Raw image of board shot by camera 4

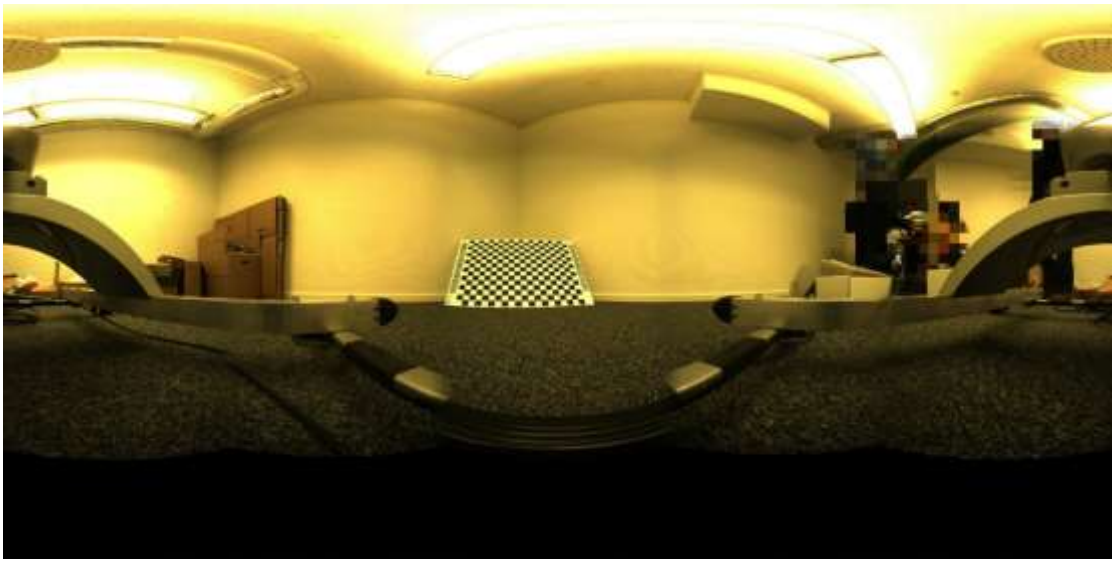

Figure A-4: Panoramic where board faces to camera 0

Figure A-5: Panoramic where board faces to camera 1

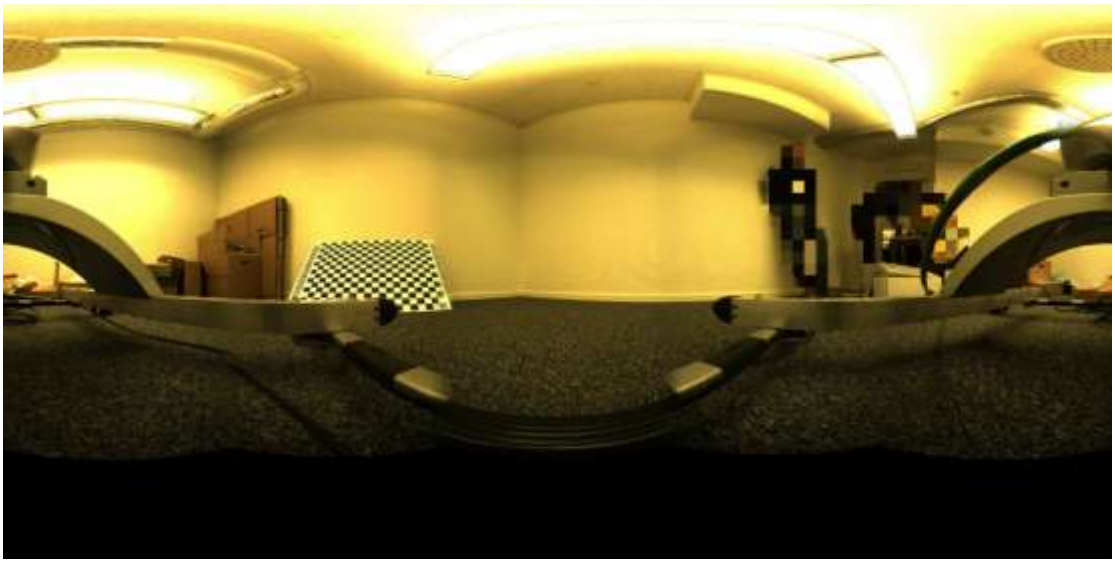

Figure A-6: Panoramic where board faces to camera 4

A.2 Point clouds

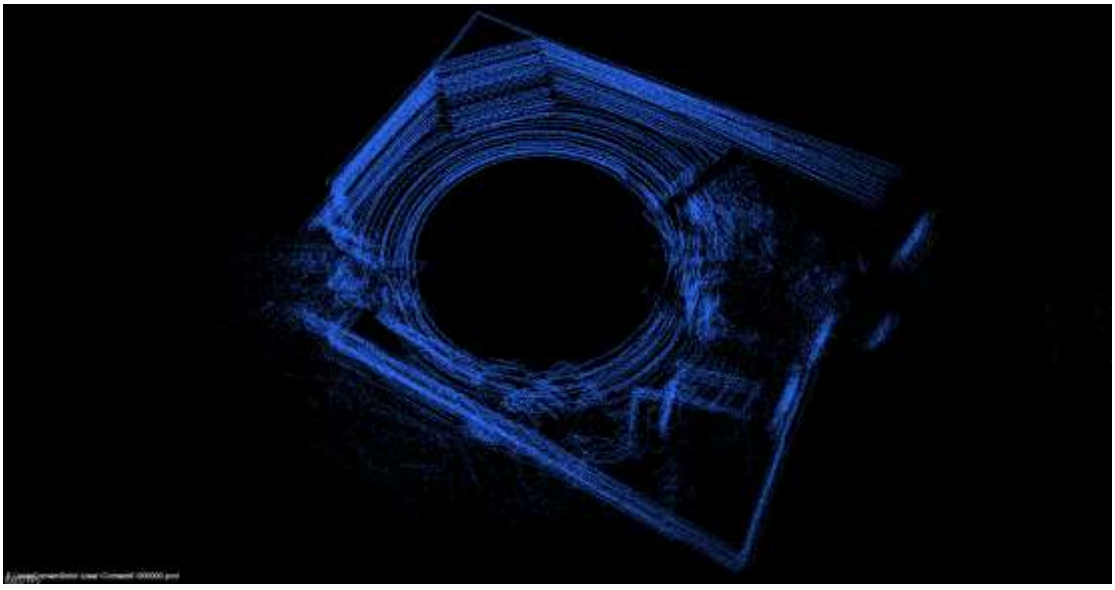

Figure A-7: Scanned room where board faces camera 0

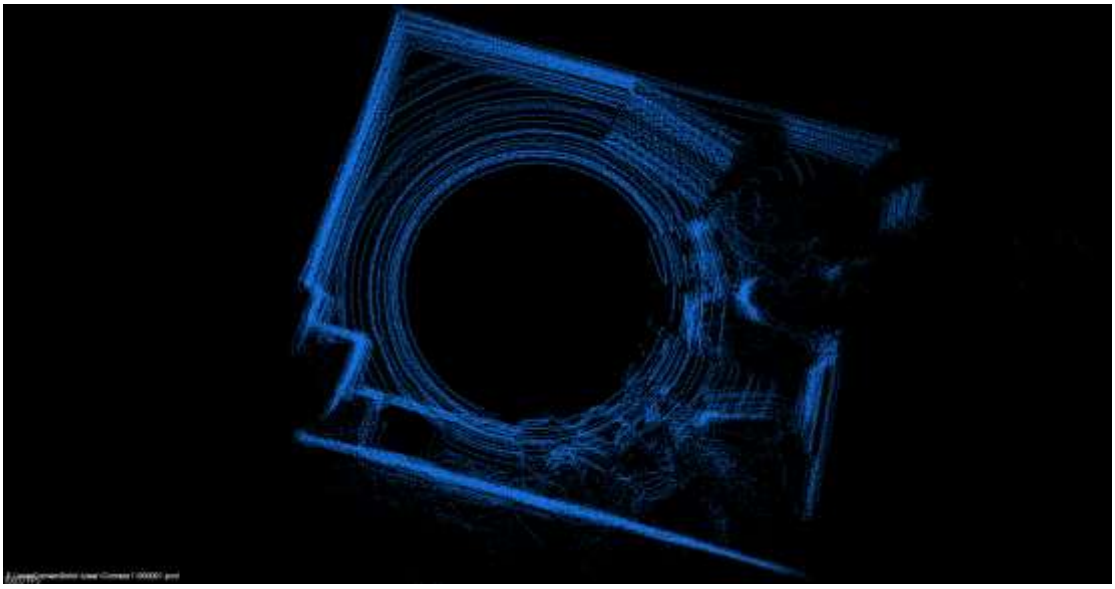

Figure A-8: Scanned room where board faces camera 1

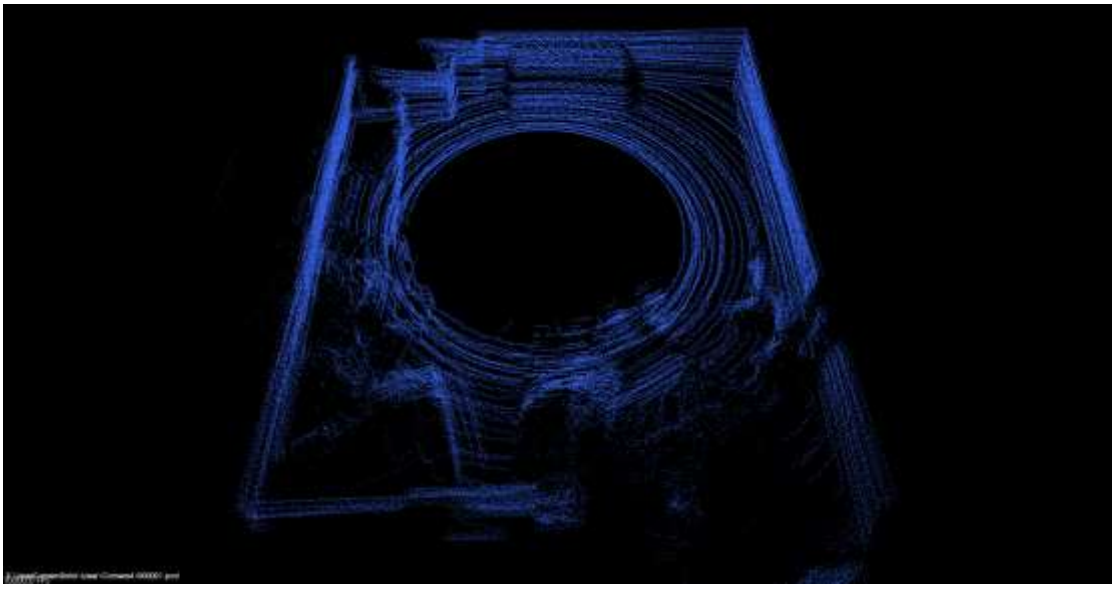

Figure A-9: Scanned room where board faces camera 4

Appendix B- Code Explanation

Brief explanation of all programming files are presented in this section.

B.1 Images related

All files that are used to process image or related to image will be demonstrated in this section.

StreamRecordingTimestamping.cpp

Requirement:

a) This program has to be run with freeglut.dll and Ladybug SDK 1.3Alpha02 or later.
b) Using Ladybug Library
c) Connected to Ladybug3 camera

Functions:

a) Record Ladybug images to PGR stream files
b) Insert timestamps to images using GPS

Input:

This program starts the first Labybug2 or Ladybug3 camera on the bus with the parameters in the .ini file defined by INI_FILENAME when it is available, otherwise, default value defined in code will be used.

Output:

Stream files with timestamps (Stream file base name is defined in ini file).

Operations:

Right click the mouse in the client area to popup a menu and select various options, or use the following hot keys:

'r' or 'R' - start recording, press again to stop recording.
'p' or 'P' - display panoramic image.
'a' or 'A' - display all-camera-image.
'd' or 'D' - display dome view image.
'Esc','q' or 'Q' - exit the program.

StreamProcessing.cpp

Requirement:

Using Ladybug Library

Functions:

a) The program can process entire or part of a stream file; an image file will be generated sequentially after each processed frame.
b) Output images of different types. Processed types including panoramic view, dome view, spherical view, rectified image can be specified either through command line or using hardcoded default value; In order to get raw image type, function 'OutputRawimage()' will be called in main method.

c) If the stream file contains GPS information, the program outputs the information to a separate text file.

Input:

Function 'processArguments(argc, argv)' can allow program to process parameters options from command line
If no parameter is passed from command line, default value will be used. Default values are hardcoded.
The program file does not use any command line temporarily.

Output:

a) Images (raw image, panoramic view, dome view, spherical view, rectified image)
b) GPS timestamps in text files

Operations:

Using Command lines to specify output path, file name, processing method and so on when Function 'processArguments(argc, argv)' is called in main method. Details are displayed in program file.

ImageCornerDetection.cpp

Requirement:

Using OpenCV library

Functions:

Detect square corners of checkerboard

Input:

Specify input through command line
a) Name of image file which contains checkerboard
b) Name of output file which contains coordinates of selected points

Output:

Text file which contains coordinates (pixels) of selected points

Operations:

Click points around the detected corner (hollow circles); Circle will become solid one it is selected.

Ladybug2Dto3DCoordinates.cpp

Requirement:

Using Ladybug library

Functions:

Transform points from 2D coordinates in image coordinate system to 3D coordinates in Ladybug head frame

Input:

Text file which contains 2D coordinates (pixels) of points

Output:

Text file which contains 3D coordinates of points in Ladybug head frame

Operations:

Specify parameters through command line

a) Name of input text file which contains 2D coordinates (pixels) of points
b) Name of output text file which contains 3D coordinates of points in Ladybug head frame
c) Sensor that is used to get the picture which pixels of points come from
d) Radius of projected spherical plane (20 is default value)

ImageBoundingBox.cpp

Requirement:

Using OpenCV library

Functions:

Find the part of image which corresponds the given set of point clouds

Input:

a) Image which contains the object corresponding to its counterpart point clouds
b) The text file which indicates the given object from point clouds

Output:

No output file, but a bounding box which encloses the object from given point louds will show up and this part of image will be cropped and displayed in another window.

Operations:

Specify following parameters through command line

a) Name of the image which contains the object corresponding to its counterpart point clouds
b) Name of the text file which indicates the given object from point clouds
c) Number of points in the text file in b)

ReadImageInfo.cpp

Requirement:

Using openCV library

Functions:

Extract color information from the input image for the points from given point clouds

Input:

a) Image
b) The text file which contains coordinates of points from point cloud

Output:

Text file which contains RGB values

Operations

Specify following parameters through command line

a) Name of the image
b) Name of the text file which contains coordinates of points from point cloud
c) Number of the points in the point cloud
d) Name of the text file which contains RGB values

Compute3DTaylorSeries.m

Requirement:

The number of points in input file should be integer times of four.

Functions:

Estimate depth of points in Ladybug head frame.

Input:

Text fine which contains coordinates of points projected on sphere in Ladybug head frame

Output:

Coordinates of points which contain depth information

Operations:

Specify following parameters for function

a) Name of the text fine which contains coordinates of points projected on sphere in Ladybug head frame
b) Name of a text file of coordinates of points which contain depth information

B.2 Point clouds

All files that are used to process image or related to point clouds will be demonstrated in this section.

ExreactPointCloud.cpp

Requirement:

Using PCL library

Functions:

Read points from point cloud and write them into text file

Input:

Point cloud file

Output:

Text file converted from point cloud file

Operations:

Specify following parameters for function

a) Name of the text fine which contains coordinates of points projected on sphere in Ladybug head frame

b) Name of a text file of coordinates of points which contain depth information

BoardExtraction.cpp

Requirement:

Using PCL library

Functions:

a) Extract checkerboard and its boarder;
b) Compute coordinates of points which correspond to those in image in Velodyne coordinate system

Input:

Point cloud which contains the object to be extracted

Output:

a) Extracted planes, hulls, lines
b) A text file which contains the square corner points

Operations:

Pass following parameters through command line-

a) Name of the image which contains the object corresponding to its counterpart point clouds
b) Base name of the extracted planes, hulls, lines
c) Name of the text file which contains the square corner points
d) Coordinates (row and column) of the square corner points which correspond to the points in image (only four points are used in this case)

B.3 Data fusion

This part tells how to relate data acquired from different sensors.

Transformation.cpp

Requirement:

Using Eigen Library

Functions:

Compute the transformation parameters between two coordinate systems using given points correspondences

Input:

a) Text file which contains points correspondences from image
b) Text file which contains points correspondences from point cloud
c) Text file which contains the point clouds to be transformed

Output:

a) Transformation parameters
b) Text file contains the transformed point clouds in Ladybug coordinate system

Operations:

Specify the parameters through command line

a) Name of the text file which contains points correspondences from image
b) Name of the text file which contains points correspondences from point cloud
c) Number of points used for computation
d) Name of the text file which contains the point clouds to be transformed
e) Name of the text file which contains transformed point clouds in Ladybug coordinate system

Ladybug3DtoLadybugPixel.cpp

Functions:

Transform 3D points in Ladybug head frame to corresponding pixels in panoramic image

Input:

Text file which contains 3D coordinates of points in Ladybug head frame

Output:

Text file which contains 2D coordinates (pixels) of points of panoramic image

Operations:

Specify the parameters through command line

a) The method to be used (input '2' in this case)
b) Number of points to be transformed
c) Name of text file which contains 3D coordinates of points in Ladybug head frame
d) Name of text file which contains 2D coordinates (pixels) of points of panoramic image

RangeColorFusion.cpp

Requirement:

Using PCL library

Functions:

a) Mapping color from image to point clouds
b) Visualizing the fused data

Input:

a) Point clouds file
b) Text file contains RGB values

Output:

Point clouds containing color information

Operations:

Specify the parameters through command line

a) Name of the Point clouds file
b) Name of the text file contains RGB values
c) Name of the point clouds file containing color information

Reports in Geodesy and Geographic Information Technology

The TRITA-GIT Series - ISSN 1653-5227

15-001 **Yanjing Jia.** Object-based Land Cover Classification with Orthophoto and LIDAR Data. Master of Science thesis in Geoinformatics. Supervisors: Hongtao Hu (KTH), Andreas Rönnberg and Mattias Pettersson (Lantmäteriet). February 2015.

15-002 **Fentaw Degie Melesse.** Evaluation of Photogrammetric Total Station. Master of Science thesis in Geodesy No. 3134. Supervisor: Milan Horemuz. April 2015.

15-003 **Tobias Jonsson.** Snake Grid och andra objektanpassade projektioner för svensk infrastruktur. Master of Science thesis in Geodesy No. 3133. Supervisors: Anna Jensen (KTH) och Ingemar Lewén (Trafikverket). May 2015.

15-004 **Joel Bergkvist**. Optimal design of network for control of total station instruments. Master of Science thesis in Geodesy No. 3135. Supervisors: Milan Horemuz (KTH) and Johan Vium Andersson (WSP). May 2015.

15-005 **Jacob Ekblad and Jacob Lips**. Development of a Workflow for Automatic Classification and Digitization of Road Objects Gathered with Mobile Mapping. Master of Science thesis in Geodesy No. 3136. Supervisors: Milan Horemuz (KTH) and Johan Vium Andersson (WSP). June 2015.

15-006 **Johnny Bärlund**. Numerical Investigation on Spherical Harmonic Synthesis and Analysis. Master of Science thesis in Geodesy No. 3137. Supervisor: Huaan Fan. June 2015.

15-007 **Nikolaos Tsalikis.** Analysis of dissemination of waterborne pathogens from raw water and related health issues in Sweden using GIS. Master of Science thesis in Geoinformatics. Supervisor: Yifang Ban. June 2015.

15-008 **Gustaf Uggla.** Large scale city models. Master of Science thesis in Geoinformatics. Supervisor: Yifang Ban. June 2015.

15-009 **Daniel Åkerman.** Adaptiv pan-skärpning av DMC bilder. Master of Science thesis in Geoinformatics. Supervisor: Hans Hauska. June 2015.

15-010 **Natallia Azaronak.** Building 3D models from geotechnical data. Master of Science thesis in Geoinformatics. Supervisor: Hans Hauska. June 2015.

15-011 **Guanyi Zhao.** Fusion of Ladybug3 omnidirectional camera and Velodyne Lidar. Master of Science thesis in Geodesy No. 3138. Supervisors: Milan Horemuz and Patrik Andersson (Volvo Cars). August 2015.

TRITA-GIT EX 15-011
ISSN 1653-5227
ISRN KTH/GIT/EX--15/011-SE