MULTI-SENSOR REGISTRATION FOR AUTONOMOUS SYSTEMS

A Dissertation

by

SUBODH MISHRA

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Srikanth Saripalli
Committee Members,	Swaminanthan Gopalswamy
	Zohaib Hasnain
	Dylan Shell
Head of Department,	Guillermo Aguilar

August 2022

Major Subject: Mechanical Engineering

ABSTRACT

This work presents novel research contributions on the topic of Multi-sensor Registration for Autonomous Systems. The research contribution can be divided into two parts. The first part (Part I : Chapters 2 - 5) presents registration of sensors (LiDARs, Cameras & IMUs) rigidly attached to a handy sensor suite and the second part (Part II : Chapters 6 & 7) presents techniques to register static cameras mounted on the infrastructure surrounding an autonomous system. In Part I, this study presents novel techniques to perform pair-wise extrinsic calibration of LiDAR-Camera pairs and LiDAR-IMU pairs, and extends the research on pair-wise sensor calibration to perform joint registration of a LiDAR, an IMU and a Camera attached to rigid sensor suite under an Extended Kalman Filter (EKF) framework that not only estimates static registration between sensors but also evolving pose of the sensor suite including a map of the environment. Thus, Part I concludes with a filtering framework to perform generalized sensor registration. In Part II, this study presents batch optimization based framework to register multiple static infrastructure sensors and also an information theoritic mutual information maximization based approach to register a single static fisheye camera in a prior map. The contributions of Part I & Part II may be brought together to build autonomous systems that leverage sensor measurements from both on-board and off-board sensor systems which improve an autonomous system's perception and state-estimation capabilities with positive downstream impacts of increased situational awareness, sensor redundancy and improved public safety.

ACKNOWLEDGMENTS

I extend my sincerest gratitude to my supervisor Dr. Srikanth Saripalli for accepting me as a graduate student in his lab. I deeply value his inputs and recommendations without which this dissertation would not have seen the light of this day. Most importantly, I want to thank him for providing me a Graduate Research Assistantship for all the semesters during my Ph.D. journey. This not only helped me focus single mindedly on research, but also gave me additional time for pursing personal interests and goals. I will be failing in my duty if I do not acknowledge his contribution in introducing me to Philip R Osteen of U.S. Army Research Laboratory (ARL) and Gaurav Pandey of Ford AV LLC, both of whom I am immensely grateful to, for collaborating with me on my Ph.D. projects. Finally, I am grateful to Dr. Saripalli for allowing me to go for internships whenever I had an opportunity. I also want to thank my committee members Dr. Dylan Shell, Dr. Zohaib Hasnain, and Dr. Swaminathan Gopalswamy for promptly responding to my requests at all times and being supportive of my work.

Outside Texas A&M University, I want to thank Dr. Gaurav Pandey, Dr. Armin Parchami and Dr. Punarjay Chakravarty, who were my mentors during my internship with Ford AV LLC. My stint at Ford was extremely productive thanks to the able supervision provided by Armin and Jay, who made sure that my internship projects were aligned with the general direction of my Ph.D. and that I have access to all Ford resources to solve the problems I was working on. Also, special thanks to Gaurav for being an important part of my Ph.D. journey, for helping me with ideas on sensor calibration, and for introducing me to folks at Ford AV LLC.

At our lab, I am thankful to George and Peng for helping me collect data to experimentally evaluate my algorithms. Outside work, I want to thank George for inviting me to BBQ parties at his place; Alvika and Peng for often accompanying me to coffee/tea. Most important of all, I am thankful to my parents and my little sister for being my pillar of strength, and encouraging me at all stages of my professional journey. I am grateful to my friends (Ananya, Chiranjeev, Sheel, Prafulla and Sumedh) for being around me whenever I missed home or felt lonesome. I could bear

the distance from home during a world pandemic only because of my dear friends. Not least of all, I want to thank everyone who make Texas A&M such an amazing place to study, have fun and conduct top class research.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of my advisor Dr. Srikanth Saripalli, and members Dr. Swaminathan Gopalswamy, Dr. Zohaib Hasnain and Dr. Dylan Shell. Dr. Saripalli, Dr. Hasnain and Dr. Gopalswamy belong to the Department of Mechanical Engineering and Dr. Shell belongs to the Department of Computer Science and Engineering.

Philip R. Osteen from U.S. Army Research Labs provided data for comparison and benchmarking in Chapter 2. Sushruth Nagesh and Sagar Manglani from Ford Motor Company provided data for experiments in Chapter 6. Armin Parchami from Ford Motor Company provided data for quantitative real world evaluation in Chapter 7.

Funding Sources

This research was financially supported by U.S. Army Research Labs and Ford Motor Company.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Introduction

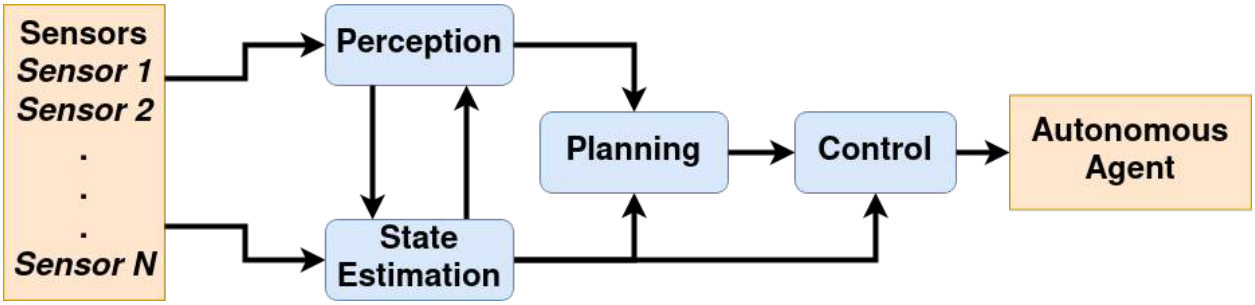


Figure 1.1: **Various components of autonomous system:** The components in blue are algorithmic/software components.

Autonomous Systems need sensors for environment perception and state estimation. The outputs from perception and state estimation modules are used in downstream planning and control tasks required for autonomous operation. A schematic of various components is shown in Figure 1.1. The immediate consumers of sensor data are the perception and state estimation algorithms, where the perception algorithms are used to perceive an agent's environment by detecting, segmenting, classifying objects, and the state estimation algorithms are used to estimate the agent's position, orientation, velocity in a given frame of reference[1]. Since different sensors measure different physical quantities and operate at different frequencies, autonomous systems usually require multi-sensor multi-modal sensing in order to collectively leverage advantages of all sensors by overcoming individual shortcomings.

For example, cameras and LiDARs help robots perceive the environment by providing complementary information. Cameras lack depth information which LiDARs provide, and LiDARs lack color, texture & appearance information which cameras provide, hence systems using both these

[1] in addition to building a map of the environment

sensing modalities can compensate each others' weaknesses by leveraging respective advantages. Cameras are better sensors when compared to LiDARs for performing object detection & classification, place recognition & detecting loop closures, but they are affected by illumination, while LiDARs are immune to illumination changes and can provide dense metric maps. Lastly, deep learning based methods are easier to apply to camera data when compared against LiDAR data for tasks like semantic segmentation and bounding box prediction. When combined with LiDAR pointcloud, these labels in the image plane can be transferred to LiDAR point cloud too. Therefore it is necessary to combine data from multiple sensors to obtain a richer representation of the surrounding world.

As far as pose/state estimation is considered, one may use GPS measurements and combine it with results from a simultaneous localization and mapping (SLAM [9], [10]) algorithm. But, monocular visual odometry/SLAM suffers from scale uncertainty and Lidar Odometry/SLAM suffers from effects of scan motion distortion owing to the spinning nature of commonly used LiDARs. In addition, GPS, cameras and LiDARs are also limited by the frequency at which they can provide robot state estimates. These limitations can be mitigated by using IMUs. IMUs not only provide measurements in metric scale but also generate measurements at considerably higher frequency when compared against cameras, LiDARs[2] or GPS. The fusion of IMU measurements with the data from LiDARs, cameras or GPS provides high frequency state (pose & velocity most commonly) estimates. IMUs also help to correct ego motion distortion in LiDAR point cloud (and rolling shutter cameras). Multisensor fusion for pose estimation and mapping overcomes the disadvantages present in single sensor state estimation methods.

In practice, modern day autonomy software stacks require the fusion of sensor measurements from a variety of sensors like wheel encoders, Inertial Measurement Units (IMUs), cameras, LiDARs (Light Detection And Ranging), RADAR (Radio Detection And Ranging), Global Positioning Systems (GPS), etc., and the most common among them are presented in detail in Section 1.1.2. The next Section (Section 1.1.1) presents a common classification of sensors used in autonomous

[2]LiDARs have high firing frequency but lower scan frequency

systems.

1.1.1 Types of sensors

Sensors used in autonomy may be broadly classified as proprioceptive sensors or exteroceptive sensors. Proprioceptive sensors sense physical quantities concerning a system's own physical state, like its velocity, acceleration, number of encoder ticks, etc. and it does not measure the system's external environment. The measurements from proprioceptive sensors need to be integrated once or twice depending on the measurement concerned, in order to arrive at meaningful pose measurements. But the results of such integration may diverge rapidly over longer time horizons owing to noise in sensor measurements whose impact amplifies when integrated.

On the otherhand, exteroceptive sensors measure the state of a system's external environment by capturing appearance or depth information in the form of images or 2D/3D-pointclouds. The measurements from exteroceptive sensors can be used to determine the pose of an autonomous system by aligning adjascent measurements, and also to perceive the environment in order to plan trajectories, detect and avoid obstacles or infer another agents' motion. Exteroceptive sensors operate at lower frequencies than proprioceptive sensors. While proprioceptive sensors provide high frequency state estimates, they may diverge rapidly, but exteroceptive sensors can help correct this divergence by making absolute pose estimation between their measurement instants. Proprioceptive sensors are used to correct effect of motion on some exteroceptive sensors by a process called egomotion compensation or deskewing of measurements. Therefore, most autonomous systems use a combination of proprioceptive and exteroceptive sensors to obtain motion compensated exteroceptive sensor measurements, and high frequency low noise state estimates.

1.1.2 Sensors used in Autonomous Systems

1.1.2.1 GPS/GNSS

Global Positioning System or GPS is a commonly used navigation and positioning method in autonomous systems that can be only used outdoors. GPS is GNSS (Global Navigational Satellite System) service provided by the US government. GNSS is a satellite constellation that provides

coded satellite signals which are processed by a GNSS receiver to calculate position, velocity (and time). Several nations like Russia, China, India, Japan, and the EU also have their own GNSS which, at the very least, provide regional or local service to cater to their security and defense needs, but the GPS service by US is the most common GNSS for civilian applications all over the world. There are several kind of GNSS receivers, but broadly GNSS provides low frequency position and velocity estimates. Dual GNSS receiver can also provide orientation/bearing information without using internal magnetometers. The disadvantage of any kind of GNSS service is that it will not work inside buildings or tunnels or close to tall buildings or under dense foliage because of multi-path reflection. Moreover, GNSS in itself provides low frequency measurements which may not be ideal for closed loop control of autonomous vehicles for path tracking/following and start point to end point navigation. In practice GNSS measurements are fused with inertial sensors like IMUs and magnetometers for providing high frequency position and velocity estimates to enable closed loop automatic control.

1.1.2.2 Cameras

Cameras are exteroceptive imaging sensors which project the 3D world on to a 2D plane. There are several type of camera projection models, the most common being the pin-hole projection model [11]. Some other projection models are fisheye and omni-directional projection models [12], [13]. The model parameters, called camera intrinsic calibration parameters, need to determined by the process of camera intrinsic calibration. The fisheye, and omni-directional cameras have very wide field of view (FoV). Depending on the manner in which the camera acquires an image of a scene, they can be classified as rolling shutter or global shutter. In rolling shutter cameras, the image pixels sense the world sequentially, line after line whereas in global shuttle cameras the image pixels sense the world at once. Owing to the sequential nature of data collection, rolling shutter cameras may suffer from ego motion distortion which is more prominent if the scene has fast moving objects or if the camera itself is moving. However, global shuttle cameras do not suffer from such motion distortion. As cameras project the 3D world into a 2D plane, they lack depth information. Also, cameras do not perform well under no or low light conditions. Cam-

eras are passive sensors which consume less power when compared to sensors like LiDARs, and most importantly they are light weight. The low power consumption and low weight makes their use suitable in aerial, ground and water vehicles, especially under situation where the platform concerned has weight constraint.

1.1.2.3 LiDARs

LiDARs can be 3D point cloud scan measuring or 2D line scan measuring exteroceptive sensors, and depending on type of measurement they are called 3D-LiDAR or 2D-LiDAR respectively. In this thesis the terms 3D-LiDARs and LiDARs are used interchangeably. LiDARs sense the environment as 3D point cloud where each point represents its Cartesian position with respect to the LiDAR's origin. In addition to the 3D coordinate it also measures the LiDAR reflectance values. Although there are several solid state LiDARs which measure the 3D point cloud in one flash, most commonly used LiDARs have a spinning mechanism which spins and sequentially aquires 3D pointcloud of the surrounding. Similar to rolling shutter camera, the spinning mechanism results in ego motion distortion if the sensor is moving. Although LiDAR measurements provide depth information and can work even when there is no illumination, it lacks appearance and color information. LiDARs are active sensors, and are more power hungry when compared to sensors like cameras and IMUs. LiDARs are heavier sensors. The power and weight consideration make their use most suitable for ground vehicles, however there are many UAVs which use LiDARs, albiet with lower flight times when compared against a similar system with camera(s) onboard.

1.1.2.4 IMUs

IMUs are sensors which measure linear acceleration and angular velocities in the body frame of reference at high frequency. The linear accelerometer readings need to be double integrated in order to obtain the position and the angular velocities need to be single integrated to obtain the orientation. However, integrating without an aiding sensor like GPS, camera or LiDAR will result in the position and orientation errors to increase manifold because IMU measurements are corrupted by non stationary measurement bias and noise, which rapidly corrupt the result of integration(s).

The advantage of IMUs are that they operate at very high frequencies and can be combined with other sensors to provide high frequency state estimates, in the process they can also correct ego motion distortion in spinning LiDARs and rolling shutter cameras. Most autonomous system grade IMUs consume low power and are light in weight, making their use possible in most kinds of robotic platforms like unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGV) and also autonomous watercrafts or underwater rovers (AUVs). The process of state estimation involving IMUs also estimates the non stationary accelerometer and gyroscope biases as state variables that need to be subtracted from IMU measurements.

1.1.3 Multi-sensor Fusion and Registration

As introduced in Section 1.1.3, multi-sensor fusion is the practice of combining sensor data from different sensors to enable multi-modal perception and multi-sensor state estimation. Multi-sensor fusion helps overcome the disadvantages of individual sensors by leveraging the advantages of multiple (complementary) sensors. Having made the case for multi-sensor fusion it is essential to state the prerequisites for enabling it.

In order to fuse data from multiple sensors, one needs to know the intrinsic model of the sensors (i.e. the intrinsic calibration), the spatial offset (extrinsic calibration/registration) and temporal offset (temporal calibration) between sensor pairs. While the process of determination of intrinsic properties is called intrinsic calibration, the process of determination of temporal and spatial offsets between sensor pairs is called cross calibration (temporal calibration + extrinsic calibration/registration).

1.1.4 Intrinsic Calibration

Although it is assumed in this thesis that the intrinsic parameters of the sensors are known *a-priori*, a brief introduction to intrinsic parameter estimation for cameras, IMUs and LiDARs is presented in this section. As far as cameras are concerned the set of intrinsic parameters depend on the type of camera model applicable to the camera being used. The most common model is the pin-

hole camera model [11] which is parameterized by a camera projection matrix $\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ $\Big((f_x, f_y)$ are x & y focal lengths, (c_x, c_y) are x & y coordinates of the optical center in the image plane, γ is the skew between axes and it is usually $0 \Big)$ and camera distortion parameters (radial + tangential) $\mathbf{D} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}$. The above parameters are meant for the pin-hole camera model, other camera models ([12], [13]) may have different parameters. However, the process of determination of these parameters is usually similar, and it involves the collection of images of a calibration target held in various different ways in the camera's field of view and finally solving a non linear least square optimization problem using known dimensions of the calibration target and its projection on the camera to determine the unknown intrinsic parameters. There are several resources available online to estimate camera intrinsics, prominent among them being from OpenCV[3] & ROS[4].

When it comes to intrinsic calibration of IMUs, the user needs to determine the noise density & random walk for both the gyroscope and accelerometer - the noise characterisitcs of the IMU biases. There are several online tools[5] [6] [7] to perform IMU intrinsic calibration, and it generally involves collection of static IMU data for a prolonged period of time ($\sim$ 2 - 4 hrs), calculation of Allan-variance plots for accelerometer and gyroscope biases using the collected data, followed by calculation of the noise characteristics.

3D-LiDAR intrinsics primarily corresponds to the beam angles of different channels of the LiDAR sensor. This information is already provided by the manufacturer but there are several methods in literature [14], [15] which re-perform LiDAR intrinsic calibration to improve the range measurements. Although intrinsic calibration is an important requirement before using any sensor, it is assumed to be known *a-priori* in this thesis.

[3] https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
[4] http://wiki.ros.org/camera_calibration
[5] https://github.com/rpng/kalibr_allan
[6] https://github.com/mintar/imu_utils
[7] https://github.com/ori-drs/allan_variance_ros

1.1.5 Cross Calibration

It is easy to appreciate the fact that sensors on a system are not co located, i.e. they are located at different places in the 3D space. In order to fuse data from multiple sensors, one needs to know the spatial and temporal offset between the sensors so that the sensor data can be represented in a common frame of reference. Most muti-sensor perception and state estimation algorithms assume that the cross-calibration between sensors is known *a-priori*. Although such assumption may be valid for well integrated sensor suites procured from original system integrators, it does not, in general, hold good in a setting where users assemble sensors from different sources on their own platforms. This has motivated research on sensor registration or sensor cross calibration, and the field has seen several contributions from various robotics labs and research groups. The problem of sensor cross calibration can be divided into two separate problems, *viz.* 1. Temporal Calibration (Section 1.1.5.1), 2. Extrinsic Calibration or Registration (Section 1.1.5.2)

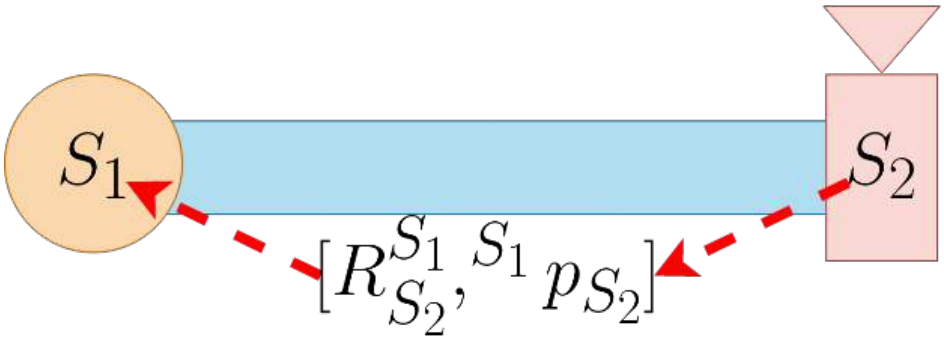


Figure 1.2: A sensor pair

1.1.5.1 Temporal Calibration

In all practical situations, a delay exists between the actual sampling of a measurement and the timestamp that is assigned to it. This delay is different for different sensors and it occurs because of the time needed for data transfer, sensor latency and operating system overhead. If one considers a

sensor pair, a time offset t_d exists between the instances at which the measurements were procured. Ideally, if this time offset is not estimated and accounted for, it may lead to unmodelled errors in the result of multi-sensor fusion. Temporal calibration can be done separately and also along with extrinsic calibration as part of a overall cross calibration routine. As far as this thesis is concerned it has been assumed that the temporal offset between sensor measurements is negligible or its effects are not consequential. This has been ensured to an extent by collecting data in a way that the effects of temporal misalignment will not be significant. For example while registering a camera with a LiDAR in Chapter 2 with the aid of a calibration target, it has been ensured that both the sensors capture measurements when the calibration target is at rest. For the cases where motion of the sensor suite is necessary, it has been ensured that the sensor suite is smoothly excited and relatively slow speeds inorder to ensure that the effect of temporal misalignment is not pertinent.

1.1.5.2 Extrinsic Calibration/Registration

Extrinsic calibration is the process of determination of 6-DoF spatial separation (rotation + translation) between sensor pairs. The rotation and the translation between sensors are usually parameterized as rotation matrix $R \in SO(3)$ and a 3D vector $p \in R^{3\times1}$ respectively. A schematic is shown in Figure 1.2 where the goal is to estimate the rotation $R^{S_1}_{S_2} \in SO(3)$ and translation ${}^{S_1}p_{S_2} \in R^{3\times1}$ between the sensors being registered or extrinsically calibrated.

The algorithm for extrinsic calibration depends on the sensor pair (proprioceptive-proprioceptive, exteroceptive-proprioceptive, exteroceptive-exteroceptive) being registered and it also depends on the physical limitations (Can the sensor system be manually motion excited?, Do the sensors on-board the suite share field of view?) of the sensor suite or the multi-sensor system. Irrespective of the sensor pair, the sensor registration or extrinsic calibration method primarily involves collection of calibration data and alignment of the acquired sensor data by enforcement of a mathematical (geometric or algebraic or both) constraint between the sensor measurements followed by solving a non linear least squared smoothing optimization or a filtering framework to estimate the unknown sensor extrinsic calibration/registration parameters.

Registration may be defined as the process of aligning sensor measurements with the goal

to determine the 6 DoF pose between them. The sensor measurements being registered may or may not be of the same modality and the alignment process consists of optimizing (minimizing/maximizing) a certain criteria or cost function (minimization of reprojection error, alignment of features detected in both modalities, alignment of motion experienced by rigidly attached sensors, etc.). Extrinsic calibration is a special case of sensor registration. By extrinsic calibration one refers to determination of a fixed/static spatial offset between a pair of sensors, while the problem of registration does not have the constraint that the sensors must be rigidly attached to each other. In general, the problem of sensor registration is the problem of estimation of pose of a particular sensor either with respect to another sensor, or with respect to the pose of the same sensor at a previous timestamp, or with respect to any fixed frame where a sensor measurement was captured. In this thesis, whenever extrinsic calibration is being discussed, the terms extrinsic calibration and sensor registration are used interchangeably and it is assumed that the intrinsic calibration and temporal calibration are known *a-priori*.

1.2 Taxonomy of Sensor Calibration/Registration Methods

As mentioned in Section 1.1.1, sensors can be broadly classified into exteroceptive sensors and proprioceptive sensors. Exteroceptive sensors (eg. cameras, LiDARs, RADARs) generate measurements by sensing the external environment, and proprioceptive sensors (eg. IMUs, wheel encoders) directly measure the state of a system [8] without depending on the environment. The method of performing registration depends on the type of sensors being registered.

Whenever the sensor suite to be calibrated consists of an proprioceptive motion sensing sensor (like an IMU or wheel encoder), one needs to perform motion excitation of the sensor suite for collecting calibration data, because the proprioceptive sensor senses nothing but motion.

If both the sensors being registered are exteroceptive and they do not share a common field of view then one needs to determine individual sensor motion and align corresponding motion segments to perform registration - also called motion based registration. However, if they share

[8] moving or at rest?, battery charge, magnetic inclination etc., for this thesis let us just concern ourselves with linear and angular motion sensing properties of proprioceptive sensors

a field of view then one can either do motion based registration or appearance based registration of the sensors. Appearance based registration method may or may not require the usage of a calibration target for easing the challenge of data association across modalities. Appearance based methods which do not use a calibration target are implemented by either aligning the dense sensor measurements or by peforming alignment of sparse salient features (like edges or planes) obtained from dense sensor measurements.

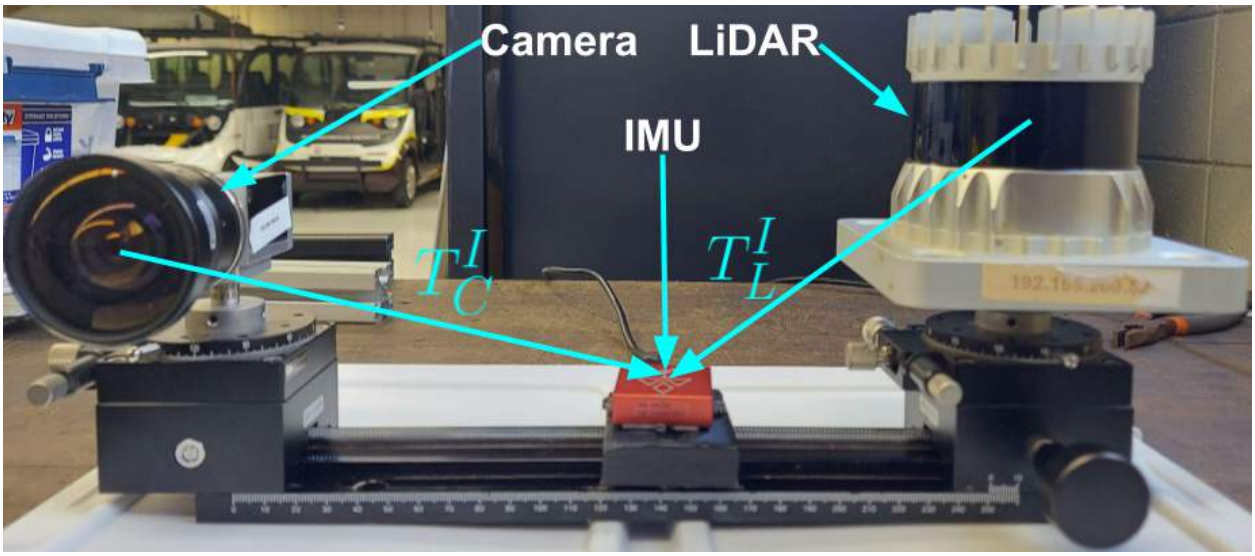


Figure 1.3: A sensor system comprising a LiDAR, a Camera and an IMU.

1.3 Contribution

This thesis addresses the problem of estimation of registration parameters ($\in SE(3)$) between pairs of sensors in a system of two or more sensors. The thesis starts with contributions on registration of sensor pairs rigidly mounted on a mobile sensor suite (like Figure 1.3), and eventually demonstrates how the methods on pairwise sensor registration motivate joint registration of multiple sensors under an Extended Kalman Filter (EKF) framework. The proposed EKF formulation not only estimates the static registration parameters/extrinsic calibration between sensor pairs but also other non static and evolving variables like sensor pose and velocities. This inspires the usage

of presented framework not only for multi-sensor extrinsic calibration but also for multi-sensor localization and mapping paving path for a generic sensor registration framework.

This thesis not only presents a filtering framework which can be easily expanded to perform multi-sensor extrinsic calibration, localization and mapping with a mobile sensor suite that can be mounted on a mobile robot like an unmanned ground vehicle or an unmanned aerial vehicle, but also presents a batch optimization formulation to register a system of several static sensors distributed over a large area. Like the previously introduced EKF formulation, the batch optimization framework also estimates non static evolving variable like a calibration robot's pose, and also a 3D map of features (tracked by the calibration robot's camera) in addition to the estimation of static registration parameters of each sensor of the distributed sensor system.

The thesis starts with novel contributions to estimate static registration parameters between sensor pairs, and what shapes in the process is an incremental development towards multiple variable filtering and smoothing sensor registration frameworks that determine poses of several static and/or moving sensors via formulation of measurement residuals using the unknown pose variables of the sensors (both static and moving) and the known measurements made by them.

1.4 Outline

The contributions of this thesis have been divided into two parts. In Part I (Chapters 2 - 5) the thesis presents methods to register sensors on a mobile sensor suite, and in Part II (Chapters 7 & 6), the thesis presents methods to register static infrastructure sensors that are be spread over a large area and are not set close to each other unlike handy sensor suite shown in Figure 1.3.

1.4.1 Outline of Part I

In Part I (Chapters 2 - 5) the thesis focuses on extrinsic calibration of LiDAR, IMU & camera sensors mounted on a mobile sensor suite with an ultimate aim to perform joint calibration of LiDAR, IMU and camera sensors together and also to demonstrate how the framework can be extened to perform online multi-sensor extrinsic calibration, localization and mapping. The thesis starts with presentation of a technique to perform target based extrinsic calibration of a 3D LiDAR-

camera pair, next it presents methods to perform pair-wise extrinsic calibration of LiDAR-IMU and camera-IMU systems and finally it combines all the discussed approaches to perform joint extrinsic calibration of LiDAR, IMU & camera together using an Extened Kalman Filter (EKF). Under the EKF formulation, the evolving pose, velocity, and biases of the IMU are also estimated.

Expanding the previous description, in Chapter 2, the thesis presents a novel target based extrinsic calibration algorithm to register a 3D LiDAR - camera pair using an easy to build planar square/rectangular calibration target. In a target based registration algorithm, the alignment of the calibration target parameters detected in both the sensing modalities by minimizing some cost function(s) results in estimation of the unknown extrinsic calibration parameters. The proposed technique detects & aligns not only the calibration target's plane/surface in both the sensing modalities but also detects & aligns its edges. The alignment of an additional feature results in superior performance (even while registering a noisy LiDAR) when compared against methods which align only the target's plane. The performance of the competing techniues is evaluated according to an independent evaluation methodology called the Mean Line Reprojection Error (MLRE) which is detailed in Chapter 2.

Next, in Chapter 3, the thesis presents a novel approach to perform extrinsic calibration of LiDAR-IMU pair. The method uses an Extended Kalman Filter (EKF) as the estimation engine for determining the spatial offset between a LiDAR and an IMU. The proposed technique is easy to use as it does not have any special requirements from the calibration environment, that is to say it does not depend on a calibration target (unlike [1]), and it does not depend on presence of dominant planar structures (unlike [1], [16]) as other competing methods do, and finally it does not need an aiding/auxilliary sensor to perform LiDAR-IMU registration (unlike [17], [18], [19], [20]). Moreover, the proposed LiDAR IMU registration algorithm does not require any tape/manually measured initialization of calibration estimates. The proposed LiDAR-IMU extrinsic calibration method has been verified, in the absence of any aiding or auxilliary sensor, using a relative verification technique where the change in calibration estimate is compared against known sensor displacement along a known direction/dimension.

In Chapter 4 the thesis compares two different alternatives to performing camera-IMU registration under an EKF framework. One in which the camera is used as pose sensor, and the other in which the camera measures pixel locations of known 3D corners on a calibration checkerboard target. The primary aim is to judge which approach gives better results so that it can be used during joint extrinsic calibration of sensors in a LiDAR-IMU-Camera system. The latter approach of using the camera as a sensor that measures the checkerboard target's corners' pixel locations, same as [8], is shown to give superior results than the case when the camera is used as a pose sensor, and therefore the author uses it in the joint registration method presented in Chapter 5.

There is no contribution in literature which performs joint registration of sensors in a LiDAR-IMU-camera system. There are methods that perform pairwise registration of LiDAR-IMU, IMU-Camera & Camera-LiDAR systems but joint estimation of spatial offsets in a joint manner for a LiDAR-IMU-Camera system is still an open search problem. This thesis presents a novel EKF based method to perform joint registration of sensors in a LiDAR-IMU-Camera system in Chapter 5. In the proposed joint extrinsic calibration method the algorithm updates LiDAR-IMU registration when a LiDAR scan is acquired. Similarly, it updates Camera-IMU registration when a checkerboard target is detected in the camera. Finally, it updates both LiDAR-IMU & Camera-IMU registration parameters simultaneously using detection of the calibration target in both camera and LiDAR. The author also qualitatively validates the superior results obtained from individual pair wise registration by projecting LiDAR points on camera image plane.

To conclude, Part I begins with registration of a Camera-LiDAR pair, proceeds to individually register LiDAR-IMU & Camera-IMU pairs, and finally utilizes the developed/implemented techniques to perform joint registration of a LiDAR-IMU-Camera sensor under an EKF framework which also estimates other variables like IMU pose and velocity thus making the proposed framework expandable to an online generic sensor registration system which can simultaneously perform sensor extrinsic calibration, localization and mapping.

1.4.2 Outline of Part II

Part II (Chapters 6 & 7) presents techniques to register static infrastructure cameras in both indoor and outdoor environments. The difference between Part I & II is that Part I concerns registration of sensors on a mobile sensor suite where the sensor suite is handy and the user can motion excite the system of sensors while Part II concerns several static sensors distributed over a large area.

In Chapter 6 the author presents a method to register several ($\sim$ 40) infrastructure cameras spread over a large ($\sim$ 150-155 m in perimeter) indoor facility. The proposed technique not only estimates static registration parameters of fixed infrastructure cameras but can also be easily extended to perform robot localization and mapping as the registration process also estimates the calibration robot's pose and a map of tracked features.

Chapter 7 of this thesis presents a method to register a wide field of view infrastructure fisheye camera in a prior map (satellite image + LiDAR mapping data) using sparse feature matching with satellite imagery for initialization of fisheye camera pose followed by maximization of mutual information between the fisheye camera image and the given LiDAR map.

Chapter 7 takes a departure from the methods of sensor registration presented in the previous Chapters in the manner that Chapter 7 aligns probability distributions of dense sensor measurements instead of geometric alignment of sparse features obtained from dense sensor data. Alignment of sparse features requires feature detection and matching between two sensor measurements which may not always be plausible because of difference in physical properties of measurement process (eg. difficulty in matching features between images of the same scene captured in a perspective camera and a fisheye camera). For instance a straight line in one sensor measurement may not look like a straight line in the other sensor. In such situations, aligning dense sensor measurements which use all the collected data to perform registration may prove to be a better solution. This is demonstrated in Chapter 7 where fisheye camera image is registered in LiDAR measurement.

Broadly, the thesis presents methods to perform generic sensor registration where both static

extrinsic calibration and evolving poses of moving sensors are determined by sparse or dense feature alignment using filtering or batch-optimization techniques.

PART I - REGISTRATION OF SENSORS ON A MOBILE SENSOR SUITE

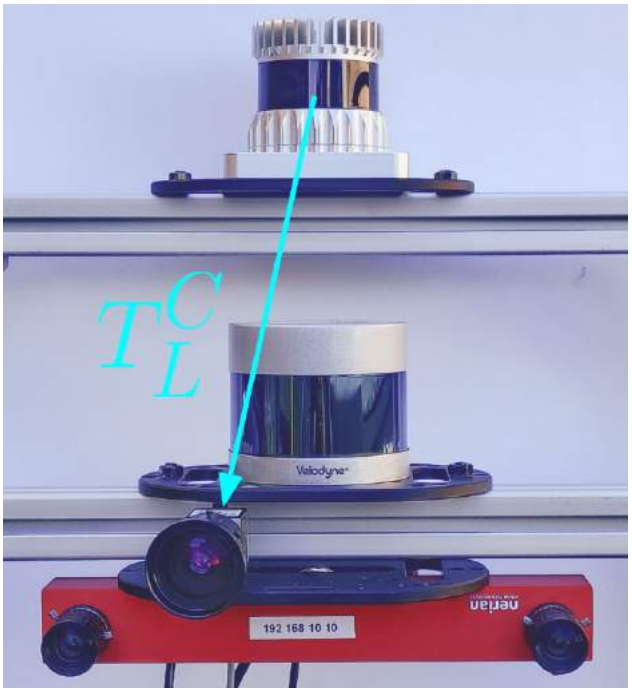


Figure 2.1: **Camera LiDAR system.** The goal of Camera LiDAR registration is to estimate $T_L^C = \begin{bmatrix} {}^C R_L & {}^C t_L \\ 0 & 1 \end{bmatrix}$, ${}^C R_L \in SO(3)$ & ${}^C t_L \in R^{3\times 1}$

2.1 Introduction

As discussed in Section 1.1, Cameras and LIDARs help robots perceive the environment by providing complementary information. Cameras lack depth information which LIDARs provide and LIDARs lack color, texture and appearance information which cameras provide, hence systems using both these sensing modalities can compensate each others' weaknesses by leveraging respective advantages. A camera can be used for recognizing objects but it doesn't tell how far an object is, such information can be obtained from a LIDAR. Cameras and LIDARs are being used together for perception, multi-sensor state estimation, (semantic) mapping and localization. Cameras are

better sensors when compared to LIDARs for place recognition and detecting loop closures but are affected by illumination, while LIDARs are immune to illumination changes and can provide dense metric maps. Simultaneous Localization and Mapping (SLAM) algorithms need both Cameras and LIDARs for robust state estimation. Hence, the knowledge of the extrinsic calibration is of paramount importance for fusing information from both these sensing modalities. Most perception and state estimation algorithms assume the extrinsic calibration to be known *a-priori*. This calls for methods to estimate the Euclidian 6 Degrees of Freedom (DoF) transformation between a LIDAR center and a Camera center using data generated by them.

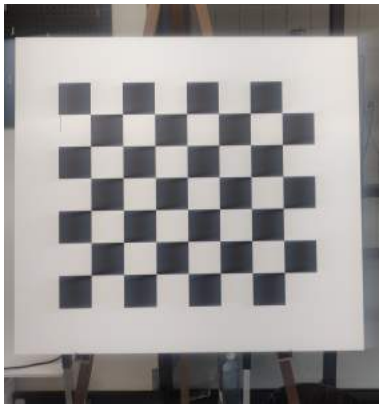
Since Cameras and LiDARs are both exteroceptive sensors and in the present context they share a common Field of View (Figures 2.1 & 2.10), as discussed in Section 1.2, appearance based methods can be used to register them. In literature, appearance based 3D-LIDAR Camera extrinsic calibration problem can be classified into two broad categories *a*) Calibration target based approaches and *b*) Calibration target less approaches. Although target-less methods show promise as far as performing online registration or online sensor calibration is concerned, they often rely on a good initial guess which usually comes from calibration target based approaches, and they are also highly sensitive to the registration environment. Given the importance of calibration target based methods for sensor registration as far as its utility in initializing target less methods and ease of use are concerned, this Chapter focuses on developing a novel target based camera LiDAR registration technique which utilizes a simple square calibration target that requires no fiducial markers on its surface unlike competing methods to perform registration of a camera LiDAR pair (Figures 2.1 & 2.10). In the works described in [21] and [22], the authors had to manually label correspondence between the camera and LIDAR. However, in the proposed technique the feature association has been done automatically.

This chapter describes how geometric relationships can be established between measurements from multimodal sensors like cameras and LiDARs, and how these relationships can be used to estimate the poses of the sensors that captured those measurements, an idea that forms the basis of any registration algorithm. The ideas described to perform Camera LiDAR sensor registration

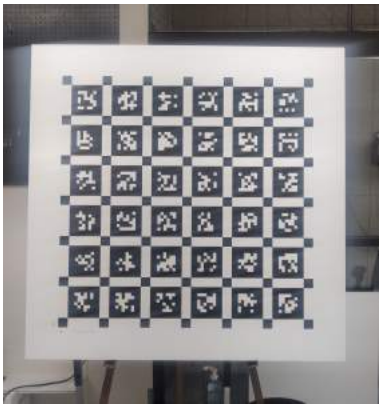
form a building block to do joint registration of multiple sensors in Chapter 5.

2.2 Literature Review

Existing 3D-LiDAR camera extrinsic calibration algorithms can be broadly classified into calibration target based [21], [23], [24], [25], [26], and calibration targetless approaches [27], [28], [22], [18]. A calibration target based approach requires a known object (like checkerboard pattern (Figure 2.2a), apriltag [29] marker (Figure 2.2b), spheres, calibration boards with circular holes (Figure 2.2c), or a room with calibration markers on all the walls as self driving car companies do) in the sensors' common Field of View (FoV) to establish geometric constraints between features detected across the sensors, whereas targetless (Figures 2.3) approaches do not require any such calibration targets. Most target based 2D/3D-LiDAR camera extrinsic calibration methods, which use one or more planar surfaces, use checkerboards or ArUco [30] or AprilTags [29] for easy detection of planar target in the camera. In cases where such markers are not used, perforated [31] & [32] or spherical targets [33] are utilized. However, as far as fabricating and detecting calibration targets in sensors is considered, planar rectangular/square targets (with or without fiducials like Aruco, Apriltags or checkerboard patterns) are the most easy to deal with.



(a) Checkerboard Target used in [21], [23], [24], [25] and [26]

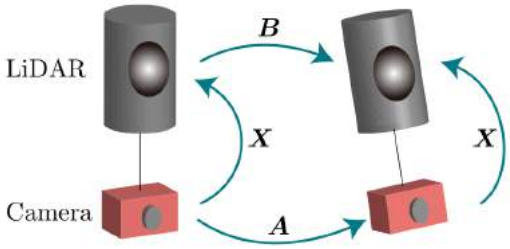


(b) Apriltag Target used in [34]

(c) Target with circular holes used in [32]

Figure 2.2: Targets used for Extrinsic Calibration of Camera-LiDAR system

(a) Aligning LiDAR pointcloud edges with edges detected on the image in [28] for performing targetless calibration



(b) Aligning LiDAR motion with Camera Motion in [35] for performing motion based targetless calibration.

Figure 2.3: Targetless calibration methods

The solution to target based 3D-LiDAR camera extrinsic calibration problem is inspired from target based 2D-LiDAR camera extrinsic calibration [36], [37], [38], [39], *etc*. The geometric constraint used in [37] is easy to use in 3D-LiDAR camera calibration scenario and has been exploited in the work presented in [21] & [23] and extended to 3D-LiDAR omnidirectional camera calibration in [24]. The idea is to detect the calibration target's plane in both the LiDAR and the Camera and solve a non linear least square minimization problem with the *point to plane* (Equation 2.2) constraint as measurement residual. [25] present a 3D-LiDAR camera calibration technique in which the rotation matrix is estimated first by aligning the detected calibration target's surface normals, and then solving a *point to plane* constraint (similar to ones in [21], [23], [24]) to determine the extrinsic calibration parameters. [26] adds more geometric constraints by introducing line correspondences in addition to the previously used plane correspondences. Line/edges of the calibration board are detected in both LiDAR and Camera, and measurement residuals are formed which involves these edge measurements and the unknown calibration parameters. The edge correspondences further constrain the optimization problem, improving the final estimate. The methods [21], [23], [24], [25] and [26] described above use a planar target with a checkerboard pattern.

[31] and [32] present calibration methods that use a rigid plane with one and four circular (Figure 2.2c) perforations respectively. The methods involve detection of circle in both the image and the 3D-LiDAR data and using the geometric constraints to determine the SE(3) transformation

between the 3D-LiDAR and the Camera. All of the aforementioned methods which use a single planar target require several observations from geometrically distinct view points. [40] presents a single view calibration technique, but uses several checkerboard planes. In addition to the *point to plane* geometric constraint that form the basis of methods described in [21], [23] and [24], the *point to back-projected plane* (Equation 2.3) constraint has been exploited in works described in [38] and [39], but only for calibrating 2D-LiDAR camera systems. The methods described thus far are pairwise LiDAR camera calibration techniques. For robots with multiple cameras and LiDARs, joint calibration techniques like [41] and [34] have been implemented. These joint calibration methods also use the *point to plane* constraint for Camera and LiDAR extrinsic calibration.

As far as target less calibration methods are considered, some of the most cited among them include [27] and [28] which depend on the external environment for calibration data. [27] maximizes the Mutual Information (MI) between camera image intensities and 3D-LiDAR reflectance values inorder to register these two sensors, whereas [28] aligns the edges in camera image to edges in 3D-LiDAR pointcloud inorder to obtain the extrinsic calibration between the camera and the 3D-LiDAR. The edges in the pointcloud data can be detected using discontinuities in range or LiDAR reflectance. The underlying assumption in [28] is that the edges detected in 3D-LiDAR are associated with the edges detected in camera image. [27] and [28] involve collection of sensor readings from several locations before solving the optimization problem, the challenges involved in these methods is that both of them require considerable computation time which will increase when the LiDARs used get denser (have more channels, hence measure more points), besides the LiDAR scans may need motion compensation if the calibration data from LiDAR sensor has motion distortion[1], In [28], one cannot always ascertain that an edge in the LiDAR data correspond to an edge in camera image, and in [27] occlusions between sensors affect the final result which requires maximization of MI over several frames. [18] and [35] are motion based target-less calibration methods which involve determining individual sensor motion and aligning them to obtain the extrinsic calibration between the sensors by using the *hand-eye-calibration* [42] constraint as

[1] which adds an additional overhead of using an IMU and determining its cross calibration w.r.t the LiDAR

measurement residual in the optimization problem. Deep Learning based methods [43], [44], [45], [46] also come under the class of appearance based target-less registration of LiDAR camera systems. Deep Learning based techniques are promising when it comes to solving the data association problem across different sensing modalities but they need to be trained with large amount of data which may require manual labelling. Although, targetless extrinsic calibration algorithms are very promising keeping online, long-term calibration tracking and correction in perspective, their estimation routine depends on good initialization which usually comes from target based calibration techniques.

This chapter proposes a novel target based calibration technique for performing 3D LiDAR-camera extrinsic calibration, and presents results along with comparison with competing techniques using real world data from several 3D LiDAR-Camera pairs. The most common approach to perform 3D LiDAR-Camera extrinsic calibration is to use a planar rectangular calibration target which can be easily and simultaneously detected (thus easing the feature correspondence problem) in both the sensing modalities. Having detected the calibration board in both the sensors an optimization problem is solved in which a cost function, formed by the detected planar target's parameters $\&$ the unknown extrinsic calibration $T_L^C \in SE(3)$ is minimized. When only the planar target features are used, the cost function is formed by residuals using the *point to plane* constraint (Section 2.4.3.1). This method has been adopted by several works [21], [23], [24], [25] for different kinds of LiDAR-camera pairs. However, in addition to the plane features, line/edge parameters of the calibration target can also be used. The work presented in [26] introduces edge/line features in addition to using the plane features and the authors demonstrate superior performance when compared against ([21]) only plane detection methods. However, [26] requires the knowledge of at least one 3D point on the calibration target's edges in the camera frame, for which one needs to make manual tape measurements w.r.t the checkerboard pattern used in the work, i.e. one needs to figure out the 3D positions of the 4 corners of the calibration board, w.r.t the origin of the checkerboard pattern, a process prone to measurement errors under normal circumstances, moreover corners of the calibration board are prone to wear and tear.

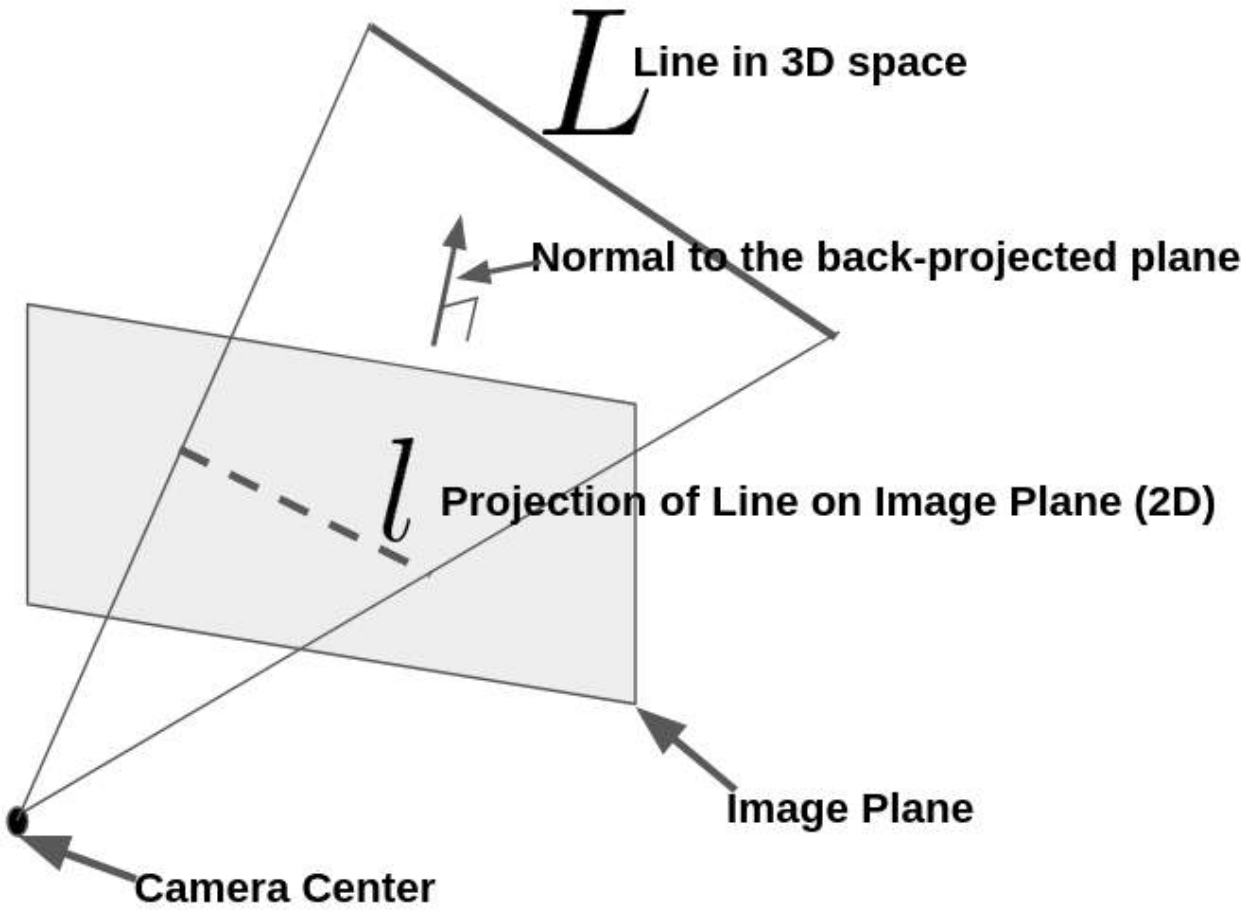


Figure 2.4: Schematic of the back-projected plane. The back-projected plane is formed by camera center and a straight line edge detected on the camera image plane.

2.3 Contributions

In this work a simple planar rectangular calibration target without any visual markers (Figure 2.11c) is used for extrinsic calibration of the 3D LiDAR-camera system. To the best of the author's knowledge, this is the first approach that calibrates a 3D-LiDAR camera system using a simple planar target without any visual marker(s) (like checkerboard or Aruco) on them. In [39], the geometrical constraint due to back-projected plane (elaborated in Section 2.4.3.2, Figure 2.4) called the *point to back-projected plane* constraint has been exploited to solve the 2D LiDAR camera extrinsic calibration problem, but no contributions could be found in literature which uses this constraint to perform registration of 3D LiDAR-camera pair. The contribution of this chapter is to

use the *point to back-projected plane* constraint to calibrate a 3D LiDAR-camera sensor suite. The *point to back-projected plane* constraint is formed by edge 3D points detected in LiDAR (Figure 2.6b) and the corresponding backprojected plane in camera. The back projected plane can be easily calculated from the calibration target's edge detection (Figure 2.4, Section 2.4.1) in camera image (Figure 2.6a). The proposed method requires the planar surface to be bounded by lines (Figure 2.6) and not curves. Although any polygonal plane can be used, a square/rectangular planar surface is used for the sake of simplicity.

2.4 Problem Formulation

For the pinhole camera model, the relationship between a 3D point P_i^L, and its image projection p_i^C, is given by

$$p_i^C = K[{}^C R_L, {}^C t_L] P_i^L. \tag{2.1}$$

The extrinsic parameters that transform the LiDAR coordinate system to that of the camera are captured by $T_L^C = \begin{bmatrix} {}^C R_L & {}^C t_L \\ 0 & 1 \end{bmatrix}$, where ${}^C R_L \in SO(3)$ is the orthonormal rotation matrix (parameterized as quaternions in the non linear optimization) and $t_L^C := [x, y, z]^\top \in R^{3\times 1}$ is the translation vector. The camera intrinsics are captured by the matrix K and is assumed to be known or estimated using Zhang's method [11]. A planar target visible in both camera and LiDAR frame is used to establish the geometric constraints that allows to estimate the rigid body transformation $({}^C R_L, {}^C t_L)$ between the two sensors (Figures 2.1 & 2.5).

2.4.1 Notations

The i^{th} observation of the planar target in camera frame is parameterized as $\pi_i^C = [n_i^C; d_i^C]$, where n_i^C and d_i^C are the normal to the target's plane and it's distance from the origin of the camera frame of reference. The 4 edges (lines) of the planar target in the image are denoted as l_{ij}^C, i is the observation number and $j = \{1:4\}$. Given the intrinsic camera calibration matrix K, the back-projected plane π_{ij}^C associated with each line l_{ij}^C is given by $\pi_{ij}^C = [K^T l_{ij}^C; 0]$ and hence, $n_{ij}^C = K^T l_{ij}^C$ [47]. To be more clear, a backprojected plane π_{ij}^C is the plane formed by the calibration

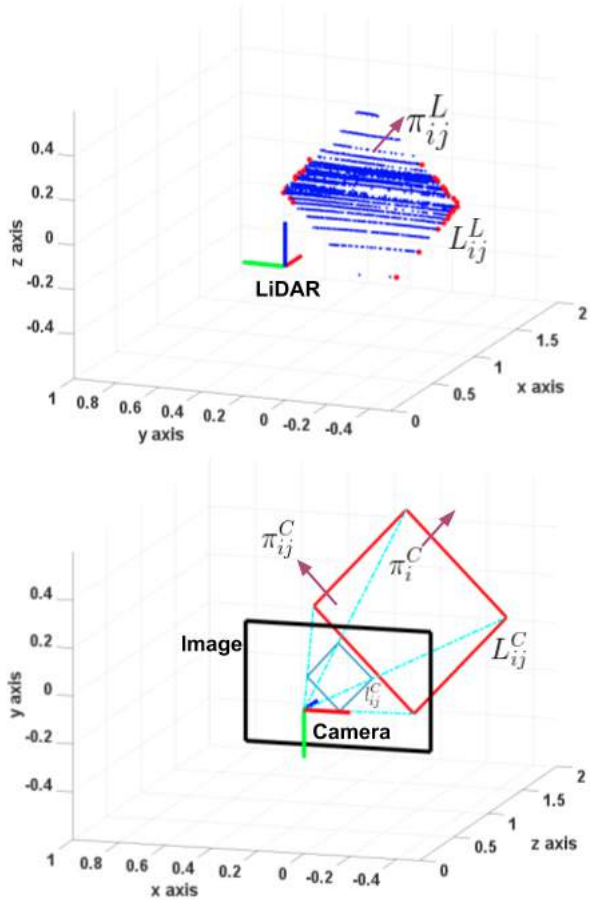


Figure 2.5: **Notations:** In the 3D LiDAR, the i^{th} pose of the planar target yields planar points $\{P_{im}^{L}\}$ and boundary points $\{Q_{ijn}^{L}\}$, where $j = \{1, 2, 3, 4\}$, which can be used to estimate π_{ij}^{L} and L_{ij}^{L} respectively. In Camera, the i^{th} pose of the planar target yields lines l_{ij}^{C}, where $j = \{1, 2, 3, 4\}$, which can be used to calculate π_{i}^{C}, π_{ij}^{C} and L_{ij}^{C}.

target's edge l_{ij} and the camera center. The corresponding 3D equation (L_{ij}^C) of the line l_{ij}^C can be determined by computing the intersection of planes π_i^C and π_{ij}^C. The 3D corners P_{ij}^C of π_i^C can be obtained by intersection of π_i^C and the back-projected planes π_{ij}^C of two adjascent lines on image plane.

For the i^{th} pose of the planar target detected in LiDAR, the normal n_i^L to π_i^L and centroid $\bar{P}_i^L$ can be calculated by using the points $\{P_{im}^L\}$ on π_i^L. Similarly, using points $\{Q_{ijn}^L\}$ on L_{ij}^L, the direction d_{ij}^L of L_{ij}^L and the centroid $\bar{Q}_{ij}^L$ can be obtained.

2.4.2 Feature Extraction

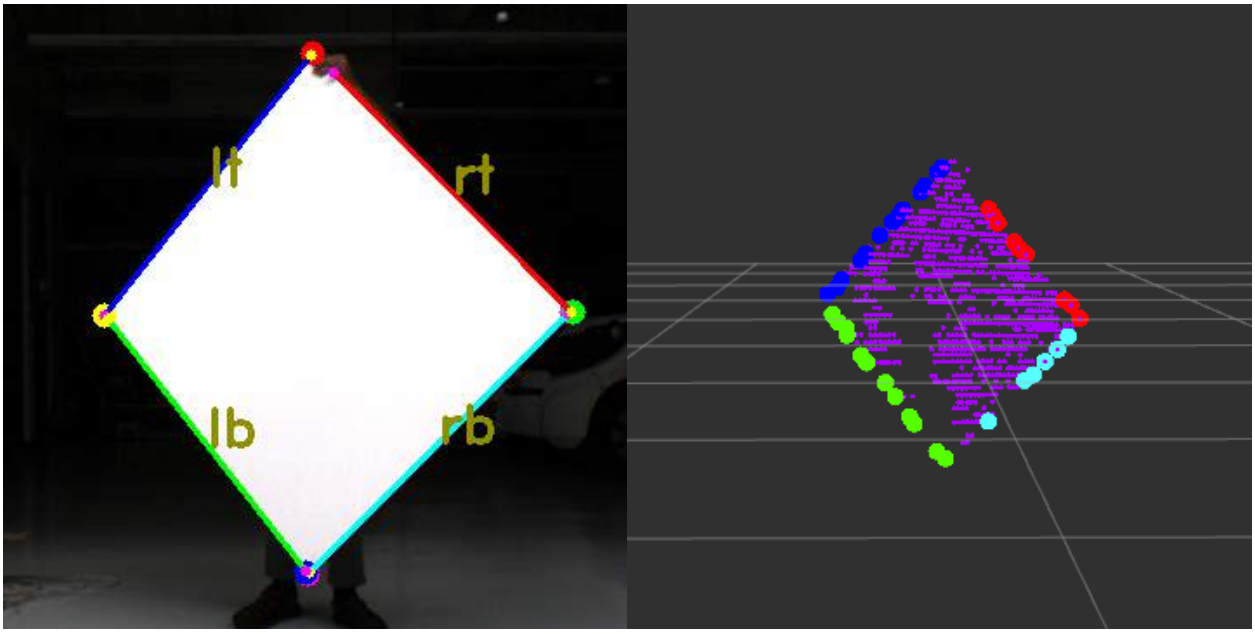
2.4.2.1 Features from Camera

Lines (l_{ij}^C) in the image are detected using Line Segment Detector [48] available in OpenCV [49]. For a four sided polygon shown in Figure 2.5, the intersection of adjacent lines l_{ij}^C provide the corner points (p_{ij}^C) in the image plane. The known dimensions of the planar target are used to solve a PnP problem and π_i^C is obtained. To summarize, the features obtained from the image are the edges l_{ij}^C (Figure 2.6) of the planar target, the corners p_{ij}^C and the plane π_i^C.

2.4.2.2 Features from 3D-LiDAR

The overview of feature detection in point cloud data is presented in Figure 2.7. Point cloud from the 3D-LiDAR is filtered to remove points which have very low probability of lying in the common FoV of the camera and the 3D-LiDAR. Next, RANSAC [50] based plane segmentation (from Point Cloud Library (PCL) [51]) is performed to obtain points $\{P_{im}^L\}$ lying on the planar target.

A three-step approach is followed to detect the points lying along the boundaries of the planar target. First, an edge detection algorithm [28] is applied to the filtered point cloud to obtain an edge point cloud. Second, to remove outliers and enforce planarity, plane segmentation is applied once again to the edge point cloud in order to obtain a point cloud formed by points which lie on the planar target's boundaries only (*viz.* boundary point cloud). Third, a RANSAC based line detection algorithm (available in PCL [51]) is applied to fit four lines to the boundary point cloud. As scan



(a) Features in Image. (b) Features in point cloud.

Figure 2.6: Result of Feature Detection in Image and LIDAR.

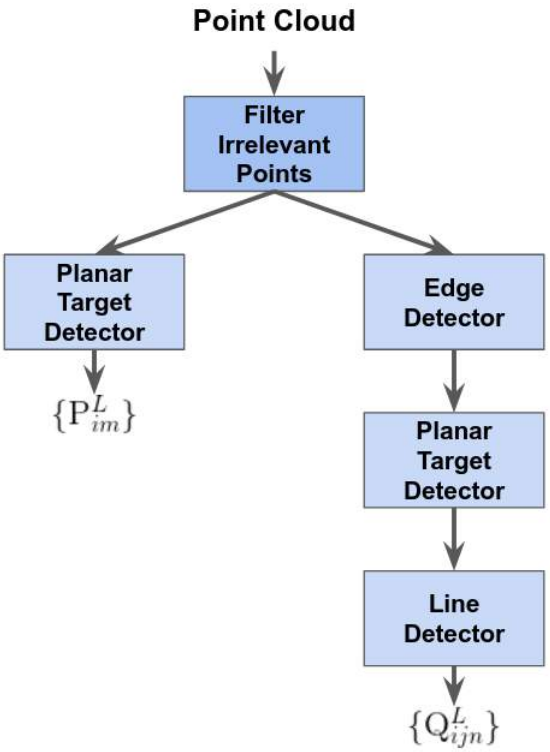


Figure 2.7: Pipeline for extraction of Planar Points $\{P^L_{im}\}$ and the points on the j^{th} edge $\{Q^L_{ijn}\}$ in LIDAR

lines are parallel to the LiDAR's horizontal plane, any edge parallel to the LiDAR's horizontal plane will be difficult to detect. Therefore edge/line detection in LiDAR is easier when the planar target is held in a way similar to Figure 2.6. In summary, the features obtained from the LiDAR data are the points on the plane $\{P_{im}^L\}$ and the points on the respective edges $\{Q_{ijn}^L\}$ of the planar target (Figure 2.6).

2.4.3 Constraints

Using the features extracted in the previous section, several geometric constraints can be established which involve the rotation ${}^C R_L$ and the translation ${}^C t_L$ between the LiDAR and Camera centers. Two such constraints pertinent to this work are discussed here in Sections 2.4.3.1 and 2.4.3.2.

2.4.3.1 Surface Point to Plane Constraint

Given the i^{th} observation of the planar target, a point P_{im}^L lying on the planar target in the LiDAR coordinate frame and the plane parameters π_i^C satisfies the geometric constraint given in Equation 2.2.

$$n_i^C.({}^C R_L P_{im}^L + {}^C t_L - d_i^C) = 0 \tag{2.2}$$

As declared earlier, $\pi_i^C = [n_i^C; d_i^C]$. Equation 2.2 is the *point-to-plane* constraint used by multiple other target based methods like [21], [23], [24], [25], [37].

2.4.3.2 Edge Point to Back-projected Plane Constraint

This is the additional constraint introduced in this thesis for 3D LiDAR - camera registration that helps the calibration parameter estimation process by further constraining the optimization. Given the i^{th} observation of the planar target, a point Q_{ijn}^L lying on the j^{th} edge of the planar target in the LiDAR coordinate frame and a corresponding line l_{ij}^C in the camera image satisfy the following geometric constraint:

$$n_{ij}^C.({}^C R_L Q_{ijn}^L + {}^C t_L) = 0 \tag{2.3}$$

Where $n_{ij}^C = K^T l_{ij}^C$ is the normal to the back-projected plane formed by the camera center and the line l_{ij} (Fig 2.5). Equations 2.2 and 2.3 are equations of the planar target and the back-projected plane respectively, in the *Hessian* normal form, in the camera frame.

2.4.4 Optimization

Cost functions for solving an optimization problem are formulated in the following ways using the geometrical constraints given in Equation 2.2 and 2.3:

2.4.4.1 Cost Function from Point to Plane Constraint

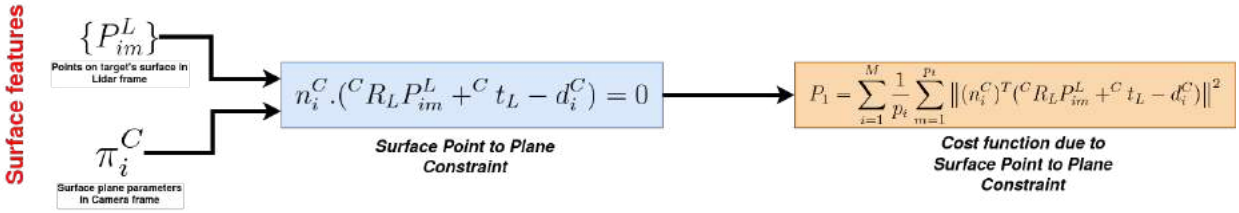


Figure 2.8: Calibration target's **surface features** give cost function $\mathbf{P_1}$. **This cost function is used in PPC-Cal, MSG-Cal & PBPC-Cal**.

The cost function formed by Point to Plane Constraint is given in Equation 2.4.

$$P_1 = \sum_{i=1}^{M} \frac{1}{p_i} \sum_{m=1}^{p_i} \left\| (n_i^C)^T ({}^C R_L P_{im}^L + {}^C t_L - d_i^C) \right\|^2 \tag{2.4}$$

Here, p_i is the number of LiDAR points lying on the planar target in the i^{th} observation and M is the total number of observations. To obtain an estimate $[{}^C\tilde{R}_L, {}^C\tilde{t}_L]$, Equation 2.4 needs to be minimized with respect to $[{}^C R_L, {}^C t_L]$.

$$[{}^C\tilde{R}_L, {}^C\tilde{t}_L] = \underset{[{}^C R_L, {}^C t_L]}{\operatorname{argmin}} \; P_1 \tag{2.5}$$

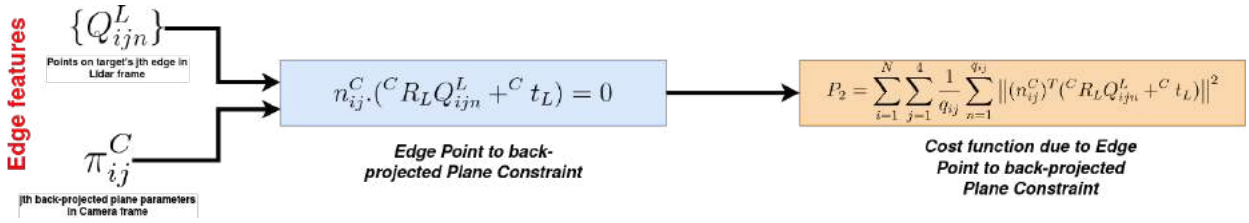


Figure 2.9: Calibration target's **edge features** give cost function $\mathbf{P_2}$. This is the cost function introduced in **PBPC-Cal**.

2.4.4.2 Cost Function from Point to Back-projected Plane Constraint

The cost function formed by Point to Back-projected Plane Constraint is given in Equation 2.6.

$$P_2 = \sum_{i=1}^{N}\sum_{j=1}^{4}\frac{1}{q_{ij}}\sum_{n=1}^{q_{ij}}\left\|(n^C_{ij})^T({}^C R_L Q^L_{ijn} + {}^C t_L)\right\|^2 \tag{2.6}$$

Here q_{ij} is the number of points lying on the j^{th} line in the i^{th} observation and N is the number of observations. To obtain an estimate $[{}^C\tilde{R}_L, {}^C\tilde{t}_L]$, Equation 2.6 needs to be minimized with respect to $[{}^C R_L, {}^C t_L]$.

$$[{}^C\tilde{R}_L, {}^C\tilde{t}_L] = \underset{[{}^C R_L, {}^C t_L]}{\operatorname{argmin}}\ P_2 \tag{2.7}$$

It has been found experimentally that the minimization problem given in Equation 2.7 is affected by the quality of initialization, especially of the rotational variables. The minimization problem given in Equation 2.5 converges to correct rotation parameters but it consistently converges to in-correct translation parameters, irrespective of the value of initialization. Therefore, Equation 2.5 is solved first and then the results obtained from solving it are used to initialize the problem given in Equation 2.7. The Ceres [5] non-linear least squares solver has been used in this work. As the usage of ***P**oint to **B**ack-Projected **P**lane* **C**onstraint is the novel contribution to extrinsic calibration of a 3D-LiDAR camera pair, this method is called **PBPC-Cal**.

Algorithm 1 Extrinsic Calibration using Plane and Line Correspondences

Input: N pairs of LIDAR and Camera Data containing the Planar Target in Field of View
Output: Rigid transformation parameters $[R_L^C, t_L^C]$ from LIDAR to Camera
Algorithm Steps

1. Collect N pairs of LiDAR and Camera Data with Planar Target in Field of View.
2. Detect the edges $\{l_{ij}^C\}$ of the Planar Target in Camera Images and use them to detect the Planar Target π_i^C in Camera Frame.
3. Detect points $\{P_{im}^L\}$ lying on the Planar Target in LIDAR data.
4. Detect points $\{Q_{ijn}^L\}$ lying on the edges of the Planar Target in LIDAR data.
5. Use the detected features to form the Cost Functions 2.4 and 2.6.
6. Solve the minimization problem given in Equation 2.5.
7. Use the solution obtained above to initialize and solve the minimization problem given in Equation 2.7.

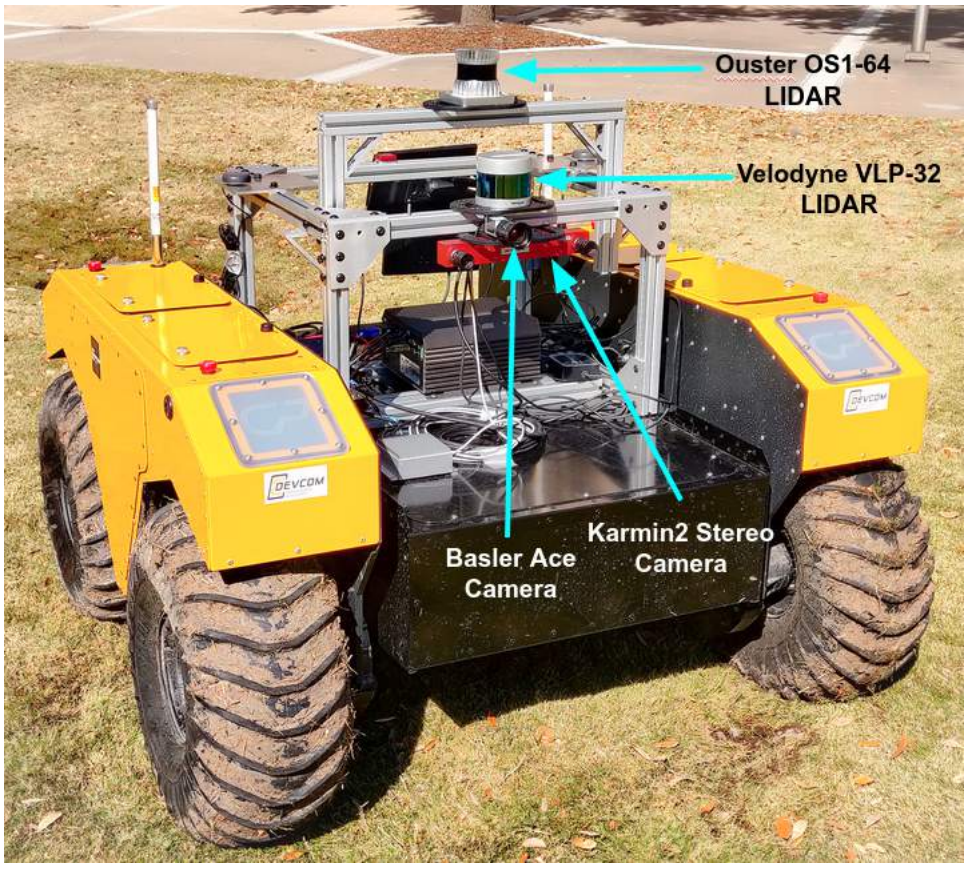


Figure 2.10: **Sensor suite on experimental platform:** Clearpath Robotics Warthog UGV with a). Ouster OS1 LIDAR, b). Velodyne VLP-32 LIDAR, c). Karmin2 Stereo Camera & d). Basler Ace Camera

2.5 System Description

The sensor suite (Figures 2.1 & 2.10) consists of an Ouster OS1 64 Channel LiDAR, a Velodyne VLP-32 LiDAR, a Basler Ace camera $[1600 \times 1200]$ and the Karmin2 Stereo Vision System (which comprises two Basler Cameras $[800 \times 600]$) such that the factory stereo calibration is known.

2.6 Data Collection

As the sensors are not time synchronized and there may be a variable latency between the generation and reception of a measurement in both the sensors, it is advisable to record data only when the planar target is at rest. A motion detection module is used which triggers calibration target detection only when the target is at rest for a user-defined duration.

2.6.1 A Note on Calibration Targets

The calibration target used in the proposed approach is a simple planar square board with no fiducial pattern like checkerboard or apriltag (Figure 2.11c) over it. The calibration target is easy to build unlike many other calibration methods which require complicated calibration targets like 3D sphere, 3D boxes, boards with circular holes, etc. It must be kept in mind that the size of the calibration board should be such that it can be easily detected in both the sensors at both near and far fields. Smaller targets may not be easily detected at far fields and very large targets may be difficult to handle by the person who collects data for calibration.

2.6.2 Data Required for Solving Point to Plane Constraint

The data required for solving the minimization problem (Equation 2.5) formed by the constraint given in Equation 2.2 are: the points lying on the planar target $\{P_{im}^{L}\}$, the normal vector from the camera to the planar surface in camera frame n_i^C and the vector from the origin of the camera frame to the origin of the planar target d_i^C in the camera frame. Data is collected by placing the Planar Target at several positions and orientations in the common FoV of the sensor suite.

The Ouster OS1 LiDAR was found to be noisier than the Velodyne VLP-32 LiDAR, this is verified by the fact that the plane segmentation algorithm returned fewer planar points when the same

RANSAC parameters were used for both the LiDARs. On an average, one receives 2600 planar points from Velodyne VLP-32 while 1200 planar points from Ouster OS1, despite the latter being a 64 channel LIDAR. Noise in calibration data affects calibration results, hence it is necessary to ensure that noisy measurements are filtered.

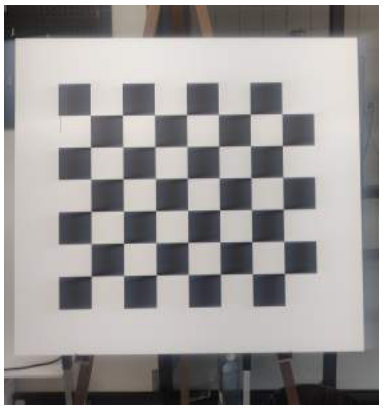
2.6.3 Data Required for Solving Point to Back-Projected Plane Constraint

The data required for solving the minimization problem (Equation 2.7) formed by the constraint given in Equation 2.3 are: the points lying on the edges of the planar target in the LiDAR frame $\{Q^L_{ijn}\}$ and the normal n^C_{ij} to the back-projected plane formed by line l^C_{ij} detected in the camera frame. While collecting data, it is essential to note that the orientation of the target is well within the thresholds which allows good edge detection. Since Velodyne VLP-32 has a non-linear distribution of scan rings about y-axis, the rings are concentrated near the origin and sparser as one pitches up or down. Therefore, the planar pattern has to be held at a suitable height so that sufficient points from its surface and edges can be collected.

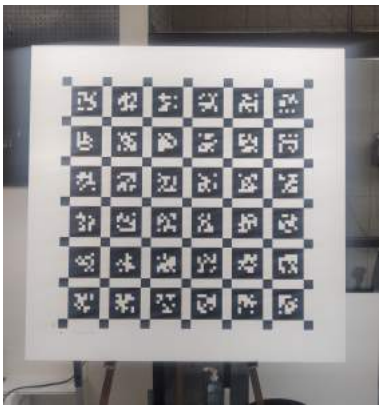
About 30 frames were collected from each sensor pair with planar target at different view points.

2.7 Experimental Evaluation

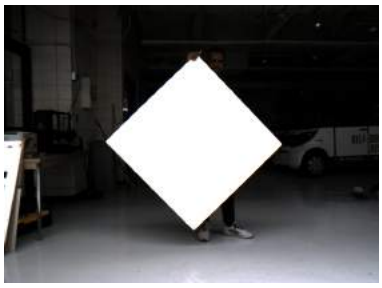
The 3D-LiDAR camera extrinsic calibration/registration algorithm proposed in this chapter (**PBPC-Cal**) has been compared with the algorithms presented in [23] (**PPC-Cal**) and in [34] (**MSG-Cal**). **PPC-Cal** and **MSG-Cal** both use *point to plane constraint* only while **PBPC-Cal** uses both *point to plane* and *point to back-projected plane* constraints. Unlike **PPC-Cal** and **PBPC-Cal** which are pair wise 3D-LiDAR camera calibration algorithms, **MSG-Cal** is a multi-sensor graph based optimization algorithm that jointly calibrates an arbitrary set of sensors (cameras and LiDARs). These algorithms have been evaluated using the sensor suite shown in Figure 2.10 to demonstrate the varying robustness of each approach to noisy sensor data. All three methods compared here use a single planar target (with known physical characteristics) as the calibration object. The proposed method **PBPC-Cal** does not require a fiducial marker on its surface for



(a) Checkerboard Target used in **PPC-Cal**



(b) Apriltag Target used in **MSG-Cal**



(c) Planar target used in the proposed method **PBPC-Cal**

Figure 2.11: Targets used for Registration of a 3D-LiDAR & a Camera for **PPC-Cal**, **MSG-Cal**, & the proposed **PBPC-Cal**.

Comparing Calibration Algorithms			
	PPC-Cal	**PBPC-Cal**[Our Method]	**MSG-Cal**
Calibration Target	Planar Checkerboard	Planar white board (of known dimensions)	Planar AprilTag board
Camera features	Checkerboard	Edges (l_{ij}^{C}) of planar board	AprilTag
3D-Lidar features	Points on planar target's surface P_{im}^{L}	Points on planar target's surface P_{im}^{L} and edges Q_{ijn}^{L}	Plane parameters $[n^T, d]^T$ derived from planar points
Minimum Views	3 non coplanar views	2 non coplanar views	Not specified
Non Linear Solver	ceres	ceres	ceres & g2o
Ease of use	Easy	Can be tedious	Easy

easy detection in camera sensor (Figure 2.11).

2.7.1 Response to Random Initialization

This section evaluates the robustness of **PPC-Cal**, **PBPC-Cal** and **MSG-Cal** to random initial conditions (Figure 2.12) drawn from a zero mean normal distribution with standard deviation of 90° and 50 cm for rotation and translation respectively. It is noticed that **PPC-Cal** and **PBPC-Cal** are robust to initialization while **MSG-Cal** exhibits divergence in a few cases where the optimization

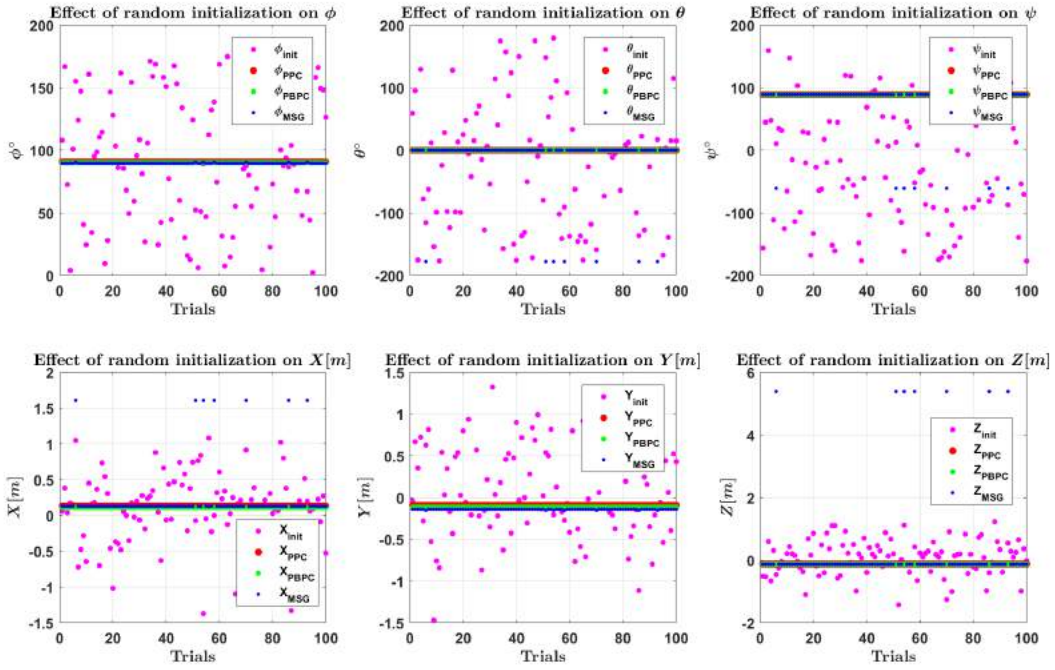


Figure 2.12: Comparing performance of **PPC-Cal**, **PBPC-Cal** and **MSG-Cal** to random initialization.

arrives at the same incorrect local minima. Besides, it is noticed that all the methods converge to nearly the same rotational values but show variation in translation values. This is because the *point to plane* constraint (which is used in all the three methods) is good at constraining rotation but it needs several observations to constrain translation. The *point to back-projected plane* constraint (used in **PBPC-Cal**) helps translation estimation accuracy by providing additional constraints at each measurement. While such bad initial guesses are not expected in practice (and initial guesses of Identity converged for each algorithm across multiple datasets) the goal is effectively to show how well each formulation constrains the optimization. Since the minimization problem(s) (for e.g. Equation 2.5, 2.7) solved to estimate $[{}^{C}R_{L}, {}^{C}t_{L}]$ are highly non-linear and involve parameters on manifolds, the convergence over several random initialization assures the user that the calibration process can be executed with any initial guess.

Errors with respect to factory stereo calibration			
	PPC-Cal	**PBPC-Cal**	**MSG-Cal**
α°_{err}	-0.0055535	0.19277	0.10103
β°_{err}	0.097271	0.19995	0.057334
γ°_{err}	-0.081701	-0.12867	-0.11242
$X_{err}\ [m]$	0.00304	0.00640	-0.00113
$Y_{err}\ [m]$	-0.00439	-0.00352	-0.00377
$Z_{err}\ [m]$	0.01124	0.00459	0.01072

Table 2.1: Errors with respect to factory stereo calibration for Velodyne VLP-32 LiDAR and the stereo rig.

2.7.2 Comparing Estimated Calibration with Factory Stereo Calibration

In the absence of ground truth the algorithms are verified by using the estimated parameters $T_L^{C_1}$ and $T_L^{C_2}$ for comparison against the known factory stereo calibration. The estimated $T_L^{C_1}$ and $T_L^{C_2}$ are compounded as $T_L^{C_1}(T_L^{C_2})^{-1}$ and compared against the given factory stereo calibration $T_{C_2}^{C_1}$. In Table 2.1 it can be seen that **PPC-Cal** and **MSG-Cal** show error in the order of 1 cm along the stereo baseline dimension (Z axis) as compared to 4.5 mm in **PBPC-Cal**. In Table 2.2, **PPC-Cal** shows better performance than the others. It can be seen that **MSG-Cal** gives the same error in both the Tables 2.1 & 2.2. It is so because **MSG-Cal** is a graph based approach which does joint optimization of all the sensors together. It is difficult to draw definitive conclusions by comparing only the stereo errors. Hence, the mean line re-projection error MLRE has been compared in Table 2.3 and Figure 2.13.

2.7.3 Mean Line Re-projection Error

Mean Line Re-projection Error (MLRE) is the average perpendicular $\perp$ distance between $\{l_{ij}^C\}$ and $\{Q_{ijn}^L\}$ projected on the image using the estimated $[{}^C R_L, {}^C t_L]$. MLRE is an independent evaluation metric since none of the methods compared in this work use it as a residual in their respective optimizations. The projection of LiDAR edge points on the image plane using the estimated extrinsic calibration parameter is shown in Figure 2.13. The results with **PBPC-Cal** have been highlighted in blue boxes and the MLRE has been tabulated in Table 2.3.

Errors with respect to factory stereo calibration			
	PPC-Cal	**PBPC-Cal**	**MSG-Cal**
α°_{err}	0.51756	0.068189	0.10103
β°_{err}	0.037753	0.12717	0.057334
γ°_{err}	-0.061076	-0.22650	-0.11242
$X_{err}\ [m]$	-0.00101	-0.00507	-0.00113
$Y_{err}\ [m]$	-0.00102	-0.00622	-0.00377
$Z_{err}\ [m]$	0.00877	0.00475	0.01072

Table 2.2: Errors with respect to factory stereo calibration for Ouster 64 Channel LiDAR and the stereo rig

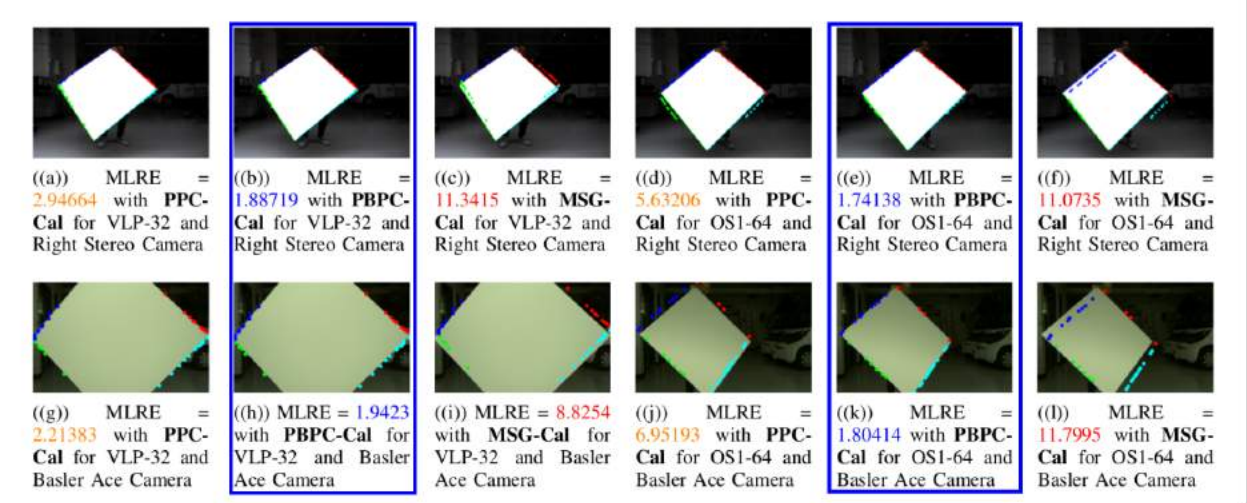


Figure 2.13: Comparing MLRE for **PPC-Cal**, **PBPC-Cal** and **MSG-Cal**

From Figure 2.13 and Table 2.3 it can be concluded that the **PBPC-Cal** performs best among all the three methods. If **PPC-Cal** and **PBPC-Cal** are compared then it can be seen that the result of **PBPC-Cal** is consistent for all the sensor pairs, as expressed by a low standard deviation (0.089039 pixels) but **PPC-Cal** shows greater variation as evident from a high standard deviation (1.9723 pixels). As discussed in Section 2.6 Ouster LiDAR is a noisy sensor. Therefore, it is seen that **PPC-Cal** performs better when used with VLP-32 LiDAR camera pair than when used to calibrate an Ouster OS1 camera pair. **MSG-Cal** which does joint optimization has all its nodes affected by Ouster's noise and therefore performs poorly as evident from a high re-projection

3D-LIDAR Camera Pair	MLRE (in pixel)		
	PPC-Cal	**PBPC-Cal** [Our method]	**MSG-Cal**
VLP-32 ↔ Stereo Left	2.57316	1.94707	11.6547
VLP-32 ↔ Stereo Right	2.94664	1.88719	11.3415
VLP-32 ↔ Basler	2.21383	1.9423	8.8254
OS1 ↔ Stereo Left	5.51552	1.76985	10.3954
OS1 ↔ Stereo Right	5.63206	1.74138	11.0735
OS1 ↔ Basler	6.95193	1.80414	11.7995
Standard Deviation	1.9723	0.089039	1.1087

Table 2.3: MLRE (in pixel) for various 3D-LIDAR Camera Pairs with **PPC-Cal**, **PBPC-Cal** and **MSG-Cal**

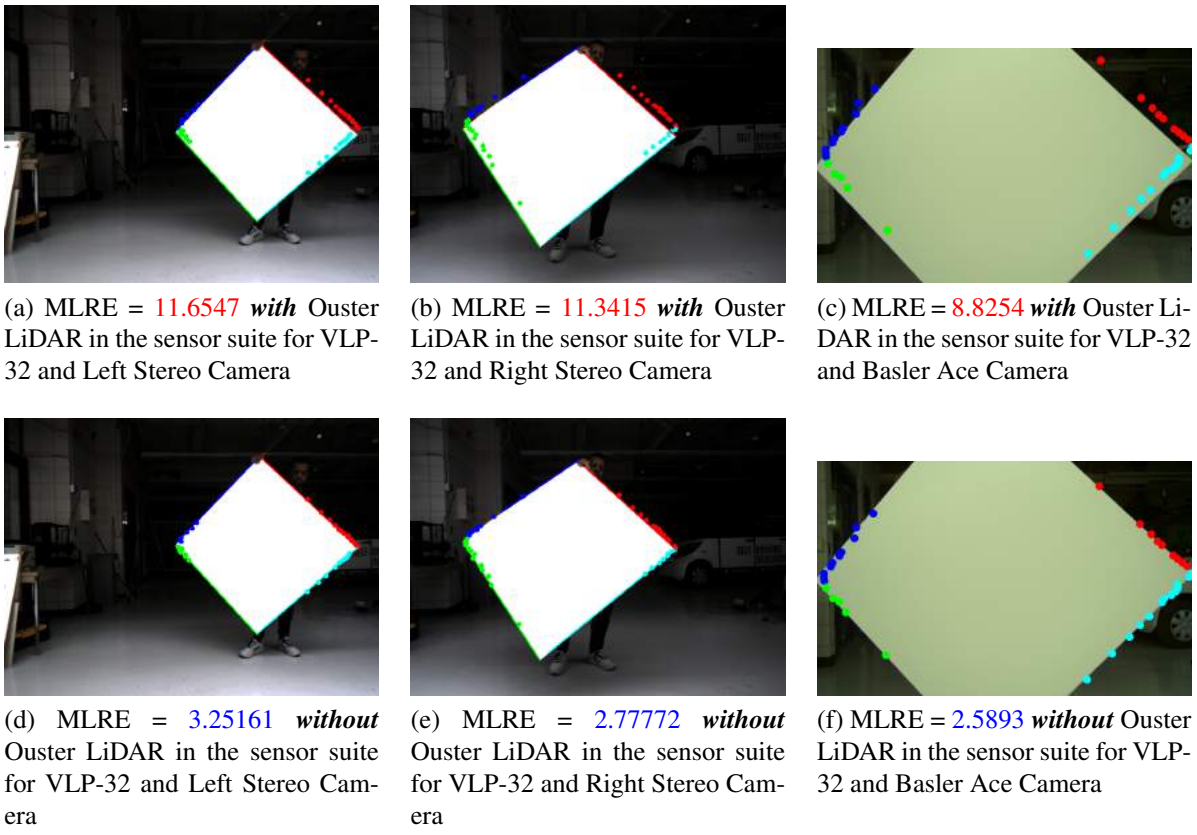
error for all sensor pairs (Table 2.3). **PBPC-Cal**, however, performs consistently better than the competing methods for all the sensor pairs.

It can be hypothesized that the graph based approach **MSG-Cal** which does joint optimization will have all its nodes affected by Ouster's noise and therefore results in a high MLRE for all sensor pairs (Table 2.3). To prove this hypothesis the sensors suite is re-calibrated using **MSG-Cal** but with the Ouster LiDAR removed, and corresponding results have been presented in Table 2.4 and Figure 2.14.

3D-LIDAR Camera Pair	MLRE		
	PPC-Cal	**PBPC-Cal**	**MSG-Cal**
VLP-32 ↔ Stereo Left	2.57316	1.94707	3.25161
VLP-32 ↔ Stereo Right	2.94664	1.88719	2.77772
VLP-32 ↔ Basler	2.21383	1.9423	2.5893

Table 2.4: MLRE (in pixels) for various 3D-LiDAR Camera Pairs with **PPC-Cal**, **PBPC-Cal** and **MSG-Cal** *without* Ouster OS1 64 LiDAR in the sensor suite.

From Figure 2.14 it can be concluded that **MSG-Cal** shows significant improvement when used in the absence of Ouster LiDAR and Table 2.4 conveys that without the Ouster in the graph optimization framework, the results of **MSG-Cal** are similar to those of **PPC-Cal** which is expected



(a) MLRE = 11.6547 ***with*** Ouster LiDAR in the sensor suite for VLP-32 and Left Stereo Camera

(b) MLRE = 11.3415 ***with*** Ouster LiDAR in the sensor suite for VLP-32 and Right Stereo Camera

(c) MLRE = 8.8254 ***with*** Ouster LiDAR in the sensor suite for VLP-32 and Basler Ace Camera

(d) MLRE = 3.25161 ***without*** Ouster LiDAR in the sensor suite for VLP-32 and Left Stereo Camera

(e) MLRE = 2.77772 ***without*** Ouster LiDAR in the sensor suite for VLP-32 and Right Stereo Camera

(f) MLRE = 2.5893 ***without*** Ouster LiDAR in the sensor suite for VLP-32 and Basler Ace Camera

Figure 2.14: Comparing performance of **MSG-Cal** ***with*** (Figure 2.14a, Figure 2.14b, Figure 2.14c) and ***without*** (Figure 2.14d, Figure 2.14e, Figure 2.14f) Ouster LiDAR in the sensor suite(Figure 2.10)

because the pair-wise calibration in **MSG-Cal** uses similar constraints as **PPC-Cal**. Irrespective of Ouster LIDAR's presence or absence, **PBPC-Cal** performs the best.

2.7.4 Time to convergence

To complete the evaluation of the three methods the execution time taken by the non linear least square solver to converge in all the three methods has been measured and the results have been presented in Table 2.5. **MSG-Cal** noticeably takes significantly less time to converge when compared against **PPC-Cal** and **PBPC-Cal**. It is so because **PPC-Cal** and **PBPC-Cal** use LiDAR points directly to form the residuals of the optimization problem while **MSG-Cal** uses parametric representations derived from those points. Therefore the number of residuals used in the minimizer

3D-LIDAR Camera Pair	Solver Time		
	PPC-Cal	**PBPC-Cal**	**MSG-Cal**
VLP-32 ↔ Stereo Left	10.7293	6.72805	0.738409996
VLP-32 ↔ Stereo Right	9.51054	8.15072	0.747992039
OS1 ↔ Stereo Left	7.56012	9.55696	0.59610796
OS1 ↔ Stereo Right	7.19404	5.07019	0.604768038
VLP-32 ↔ Basler	12.7641	10.9075	1.043144942
OS1 ↔ Basler	8.63144	3.27802	0.847215891

Table 2.5: Time taken (s) by solver to converge for various 3D-LiDAR Camera Pairs with **PPC-Cal**, **PBPC-Cal** and **MSG-Cal**

is significantly larger in **PPC-Cal** and **PBPC-Cal** and therefore they take longer to converge.

2.8 Discussion

This Chapter introduced a novel 3D LiDAR-Camera extrinsic calibration algorithm where it is shown how two different geometric constraints can be used together to perform registration of a 3D-LIDAR Camera system. The proposed method (**PBPC-Cal**) is compared with other LiDAR Camera extrinsic calibration algorithms *viz.* **PPC-Cal** ([21], [23] & [24]) and **MSG-Cal** [34]. These methods have been extensively evaluated on a multi-sensor platform (Figure 2.10) comprising 3 distinct cameras and 2 different LiDARs. From the experiments conducted, it can be concluded that **PPC-Cal** & **PBPC-Cal** are robust to random initialization for all trials while **MSG-Cal** diverged in a few trials (Figure 2.12). Nevertheless, barring a few cases in **MSG-Cal**, all the three frameworks can be initialized with any intial condition and will still converge. It's shown that **PPC-Cal** which uses only *point to plane* constraint shows deterioration in performance when a noisy sensor is used (Table 2.3). The use of additional *point to back-projected* plane constraint in **PBPC-Cal** helps reduce the effect of noisy sensor (Ouster LiDAR) by introducing more geometrical constraints to the non-linear cost function. It's also shown that the global graph based optimization method **MSG-Cal**, which uses a variant of the *point to plane* constraint has all final pair wise calibrations (as evident from high MLRE from Table 2.3) affected in the presence of a noisy sensor but gives comparable performance to **PPC-Cal** when the noisy sensor is removed

(Table 2.4). **PBPC-Cal** exhibits similar performance both with and without the noisy sensor and performs better than both **PPC-Cal** and **MSG-Cal** under all circumstances (Tables 2.3 & 2.4, Figure 2.13). If the sensor suite does not have a noisy sensor and need a quick calibration result then using **PPC-Cal** is a good option but if the sensor suite has multiple sensors (with low noise) then **MSG-Cal** should be the algorithm of choice, as collecting data for both these methods is easier. Instead, if the sensors used are noisy, then **PBPC-Cal** should be used because it has proven to work equally well under all circumstances.

2.9 Conclusion

In conclusion, the present Chapter showed how the proposed approach **PBPC-Cal** performs the best for different Camera-LiDAR sensor pairs (Table 2.3) as evident from lowest MLRE in the verification experiments, even when a noisy sensor is being registered, using only a single easy to build calibration target that does not even require a fiducial marker on it, thus making it a better alternative to other competing target based calibration algorithms. This Chapter proposed a novel Camera-LiDAR registration algorithm which aligns LiDAR points on calibration target's surface and edges with the corresponding detections in camera to determine the unknown registration between the two sensor by sequentially minimizing two different cost functions. The proposed method does not require any manual or tape measurement initialization for the optimization to converge to meaningful estimates.

As far as the big picture and a broader context is concerned, this chapter presented how measurement of a common object made by different sensors helped to determine the respective sensor poses (or the pose between them in the case of a sensor pair as is the current context). Theoretically, this technique may be extended to estimating poses of multiple sensors which need not be rigidly attached to each other but mutually tracking same features during the process of their operation. Another important aspect of the chapter is that it presents how a using different types of geometric relationships constrains the estimation pipeline more tightly and leads to convergence to values which are much closer to the truth than just using one type of geometric relationship. This idea can be extened to multi sensor SLAM pipelines which detect a variety of features like lines,

planes and curvatures in the sensors' field of view to solve sensor pose estimation problem with the claim that detecting all the features and using them for estimation is better than just detecting one kind of feature in the estimation pipeline.

As outlined in Section 1.4, the next Chapter presents a novel algorithm to perform extrinsic calibration of a LiDAR-IMU sensor suite.

3.1 Introduction

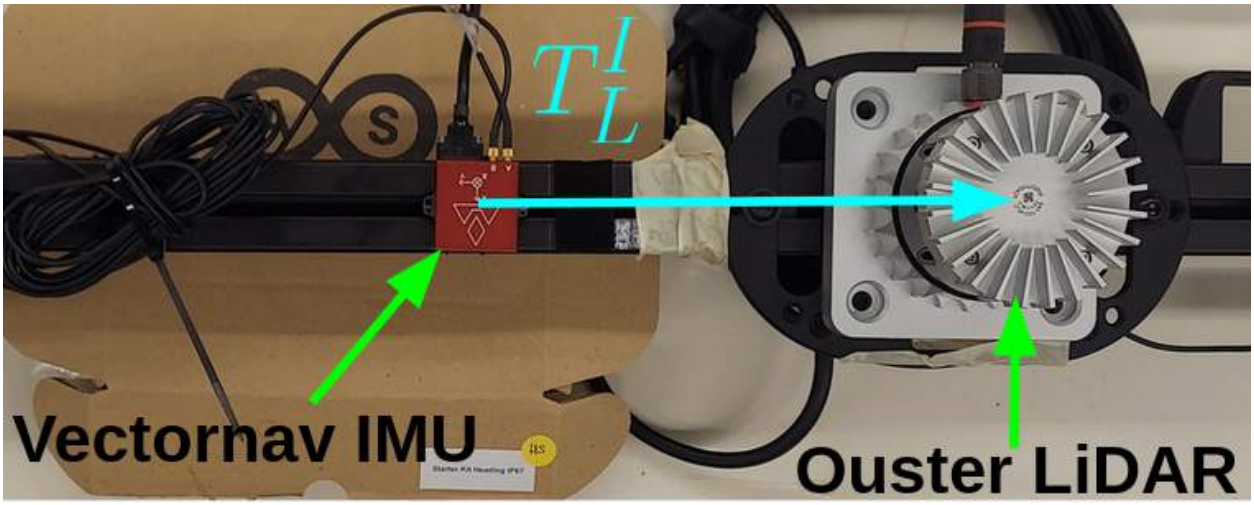
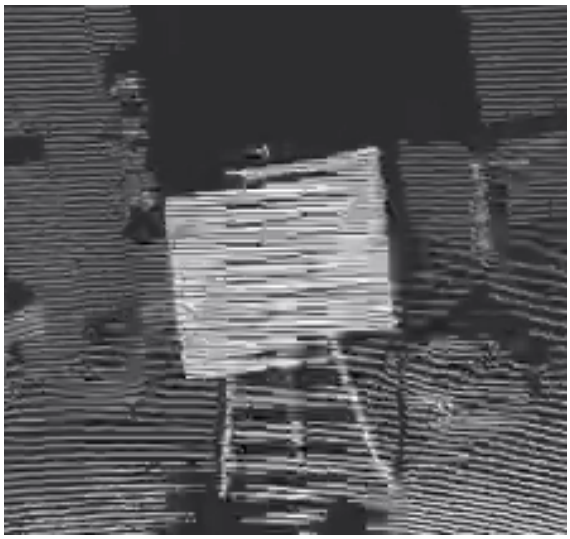


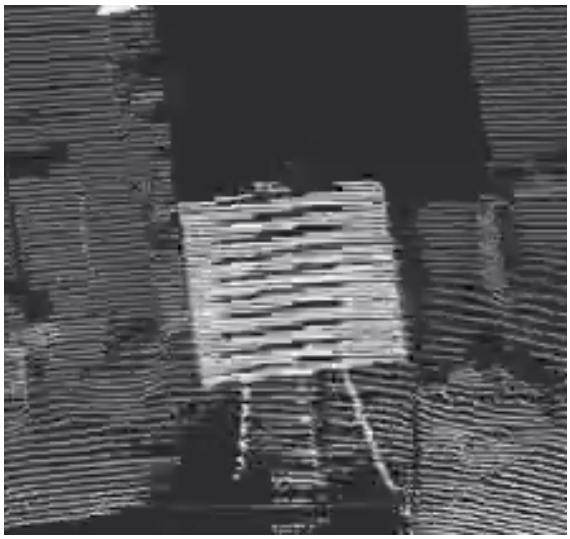
Figure 3.1: **LiDAR-IMU system.** The goal is to estimate $T_L^I = \begin{bmatrix} R_L^I & p_L^I \\ 0 & 1 \end{bmatrix}$

3D-LiDARs and IMUs are ubiquitous to autonomous robots. 3D-LiDARs provide a 3D point cloud of the area that the robot operates in, and they are not affected by illumination. This has proved the usage of LiDARs beneficial in several robotic applications. However, owing to the spinning nature of the sensor and the sequential manner in which they produce measurements, LiDARs suffer from significant motion distortion (Figure 3.2) when a robot exhibits dynamic maneuvers. The motion distortion can be seen in highway operating speeds for self driving cars and also in other robotic applications like autonomous flight and off-road robotics. Motion distorted scans deteriorate the result of LiDAR Odometry/Simultaneous Localizaton & Mapping (SLAM) algorithms (e.g. [2], [52], [53]. [54]).

Inertial Measurement Units (IMUs) can be used to mitigate the effect of motion on spinning LiDARs (Figure 3.2b). IMUs measure linear acceleration and angular velocity at frequencies higher



(a) Distorted due to robot motion. Square cardboard looks deformed/sheared.



(b) Un-distorted/Deskewed using IMU measurements and the Extrinsic Calibration T_L^I. The square cardboard looks intact.

Figure 3.2: Distorted vs Deskewed Point Cloud. Ego motion distorted point cloud restored using IMU measurements and the knowledge of sensor registration parameters, i.e. the extrinsic calibration T_L^I between the LiDAR and the IMU sensors.

than the spinning rate of a LiDARs. State of the art LiDAR Odometry/SLAM algorithms use IMUs to correct motion distortion (also called deskewing) in LiDAR scans and produce better estimates of the robot's position and the surrounding map. In order to use the IMU's measurements, these algorithms require that the spatial separation or the extrinsic calibration between the IMU and LiDAR be known *a-priori*, such that data from both these sensors can be expressed in a common frame of reference. In robotics labs where researchers generally assemble sensor suites using sensors procured from different sources, the extrinsic calibration between a LiDAR and an IMU is usually unkown. Therefore, it is important to estimate the extrinsic calibration in order to use any LiDAR Inertial Odometry or SLAM algorithm. Although IMUs operate at frequencies much higher than the spinning rate of LiDARs, the number of times a LiDAR fires a beam to acquire measurements during a 360 $^\circ$ scan (*viz,* the firing rate) is significantly higher than the IMU frequency. This requires the use of interpolation/extrapolation techniques (Figure 3.3) to match IMU rates to LiDAR firing rates so that motion compensation can be done using IMU measurements.

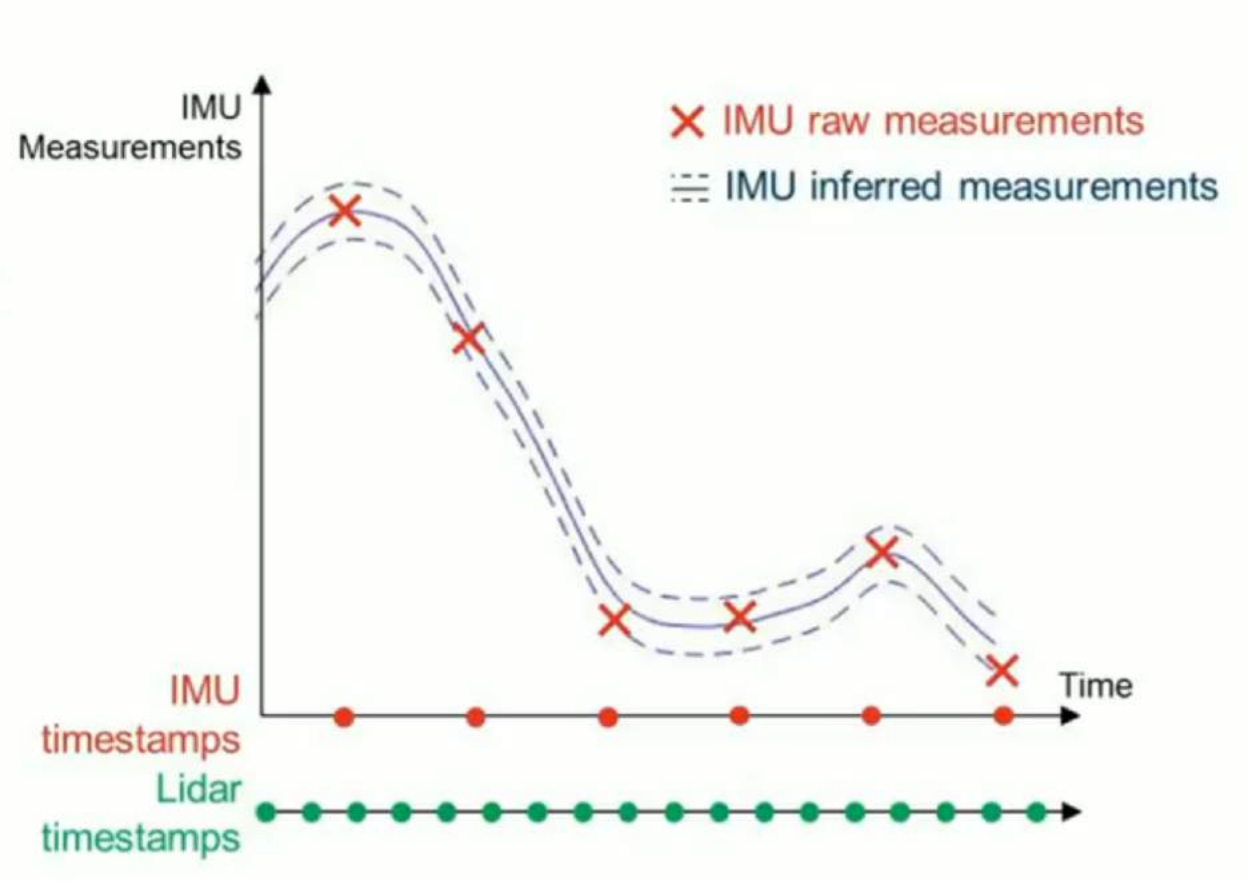


Figure 3.3: Inferring IMU measurements at LiDAR point/firing timestamps. Taken from [1]

To give some context, the Ouster 128 channel LiDAR shown in Figure 3.1 acquires one 360◦ scan in 0.1 second, however in order to acquire one scan it fires 1024 times during the course of a 360 ◦ rotation, which means that the LiDAR fires 10240 times in 1 second making the LiDAR firing frequency 10240 Hz. Hence the IMU measurements coming at lower frequency of 400 Hz need to be upsampled/interpolated/extrapolated to match the LiDAR firing timestamps so that LiDAR pose at the LiDAR firing timestamps can be inferred in-order to remove effect of motion distortion from LiDAR scans.

[1] uses Gaussian process regression to upsample IMU measurements (Figure 3.3) so that an IMU measurement corresponding to a LiDAR firing time can be regressed , [16] uses continuous time spline to model IMU state (pose + velocity) such that the spline can be differentiated and an linear acceleration/angular velocity value can be obtained for a given LiDAR firing time, and

this chapter proposes a discrete time IMU state propagation model under an EKF framework to compensate for the effect of motion during the calibration process.

3.2 Literature Review

There are numerous published works on 3D-LiDAR IMU calibration *viz.* [17], [18], [19], [20], [1], [16]. However, among these, [17], [18], [19] are methods which use GPS/GNSS information (as aiding/auxilliary sensor) for accurate pose estimation of the inertial sensor. Therefore, these approaches cannot be used when GPS is unavailable. [20] does camera IMU calibration first and then cross calibrates the visual inertial system with the LiDAR. [1], [16] do not depend on auxilary sensors for 3D-LiDAR IMU extrinsic calibration, thus making them easier to use in cases when an auxiliary sensor is unavailable.

[1] uses Gaussian Process regression [55] to up-sample IMU measurements so that a corresponding IMU reading can be inferred for every 3D-LiDAR firing time. Upsampling of IMU measurements helps remove motion distortion from LiDAR scans which occurs during the data collection phase of LiDAR IMU calibration. They use 3 orthogonal planes as a calibration reference map/target and utilize the projection of each measured LiDAR point on the corresponding plane as a geometrical constraint which requires the knowledge of the unkown extrinsic calibration parameters. They perform IMU-pre-integration [56] on the up-sampled IMU measurements and solve a non linear batch estimation problem using factor graphs to determine the unknown extrinsic calibration parameters. Although the method is promising when it comes to estimating sensor poses in an offline fashine, using GP regression along with batch optimization makes the process not suitable of online estimation of sensor registration.

[16] on the other hand models the IMU state (pose, velocity, biases), instead of IMU measurements (like [1]), as a continuous time spline which can be differentiated and equated to IMU measurements. Modelling the IMU trajectory as a continuous time spline helps infer IMU pose at LiDAR firing times. Unlike the previous method (i.e. [1]), this method exploits all the planes in the calibration environment and similar to the previous method this method also uses the projection of each measured LiDAR point on the corresponding plane as a geometrical constraint which re-

quires the knowledge of the unknown extrinsic calibration parameters. The geometrical constraint is squared and added to form a cost function which is minimized using the method of non linear least squares. Although [16] does not use any calibration target, it nonetheless requires an environment that needs to have several well defined planar structures. Similar to [1], this too is a batch optmization process which is not suitable for online longterm registration applications.

3.3 Contribution

The method presented in this Chapter, utilizes an Extended Kalman Filter (EKF) for 3D LiDAR IMU extrinsic calibration, draws inspiration from the camera IMU extrinsic calibration algorithm presented in [8]. In contrast to [8], a measurement model suitable for LiDAR has been used where the LiDAR is used as a pose sensor, generating LiDAR pose in a fixed frame of reference using scan to scan and scan to map matching of point cloud data. In contrast to [1] and [16], which use Gaussian process regression and splines for interpolating IMU measurements and IMU state respectively, this Chapter utilizes a discrete time IMU state propagation model (Equations 3.8 - 3.14) under an EKF framework to predict IMU state. The IMU state prediction model is also used to remove motion distortion from LiDAR scans.

Extrinsic calibration is required to remove motion distortion and motion distortion occurs while collecting data to do extrinsic calibration. The EKF framework allows one to use the best available estimate of extrinsic calibration parameters to motion compensate the LiDAR scans during the extrinsic calibration process. The proposed approach follows a *predict* (Section 3.5.3.1) $\longrightarrow$ *deskew* (Section 3.5.3.2) $\longrightarrow$ *update* (Section 3.5.3.4) cycle (Figure 3.5b), with a scan matching technique (Section 3.5.3.3) that uses the deskewed scans to produce measurements for state/covariance update (Figure 3.5b) in order to do extrinsic calibration of a 3D-LiDAR IMU system. The OpenVINS[57] framework is used to develop the proposed LiDAR-IMU registration/extrinsic calibration code. The proposed EKF approach is much faster than the GP and spline based technique presented in [1] and [16] respectively, especially when the time spent on solving the optimization problem is concerned.

The proposed approach does not require any calibration target (unlike [1]) or specific environ-

mental features like planes (unlike both [1] and [16]) for doing extrinsic calibration of a 3D-LiDAR and an IMU as it uses the entire scan for constraining pose between two adjascent scan instants unlike the other methods which extract a calibration target between two adjascent scans to constrain the variables to be estimated. The proposed 3D-LiDAR IMU extrinsic calibration algorithm does not depend on usage of auxiliary pose sensors like GPS/GNSS or camera-inertial sensor suites, which [18], [17], [19], [20] do. The proposed method does not require any tape measured initialization of the calibration parameters (unlike [8]). A generic Error State EKF for Visual Inertial Navigation presented in [57] is utilized and repurposed to incorporate the *motion based calibration* (Section 3.4) constraint utilized in [18] to formulate a novel 3D-LiDAR IMU extrinsic calibration algorithm. The proposed method also demonstrates the estimation of non static IMU pose+velocity in addition to the estimation of static registration between LiDAR and IMU. Finally, this contribution is the second open-sourced 3D-LiDAR IMU calibration algorithm[1] which does not depend on any auxiliary sensor, with [16] being the first.

3.4 Motion Based Extrinsic Calibration

The motion based constraint for corresponding sensor motion is given in Equation 3.1 and a schematic is shown in Figure 3.4. This constraint is similar to the standard Hand Eye Calibration [42] constraint commonly used in calibration of robotic manipulators and extended to calibration of multimodal mobile robotic sensors in [18]. In the current scenario, $T_{L_k}^{L_{k-1}}$ is the motion between two LiDAR scans which can be obtained by using any scan matching technique - like NDT [58] and $T_{I_k}^{I_{k-1}}$ is the motion experienced by the IMU between the two scan instants. For the sensors spatially separated from each other by a fixed rigid pose T_L^I (the extrinsic calibration parameter), the motion based calibration constraint is given in Equation 3.1.

$$T_{I_k}^{I_{k-1}} T_L^I = T_L^I T_{L_k}^{L_{k-1}} \tag{3.1}$$

[1] `https://github.com/unmannedlab/imu_lidar_calibration`

Here, $T_{I_k}^{I_{k-1}} = \begin{bmatrix} R_{I_k}^{I_{k-1}} & {}^{I_{k-1}}p_{I_k} \\ 0 & 1 \end{bmatrix}$ $T_{L_k}^{L_{k-1}} = \begin{bmatrix} R_{L_k}^{L_{k-1}} & {}^{L_{k-1}}p_{L_k} \\ 0 & 1 \end{bmatrix}$ and $T_L^I = \begin{bmatrix} R_L^I & {}^{I}p_L \\ 0 & 1 \end{bmatrix}$.
$R_b^a \in SO(3)$ is a rotation matrix and ${}^{a}p_b \in R^{3\times1}$ is a translation vector. The motion based constraint given in Equation 3.1 is used for estimating the inter sensor rotation $R_L^I \in SO(3)$ and translation ${}^{I}p_L \in R^3$. Specifically, the rotation component of Equation 3.1 is utilized to initialize the inter sensor rotation and this initialization is used to estimate the inter sensor translation using an Extended Kalman Filter (EKF) based estimation algorithm [8], [57].

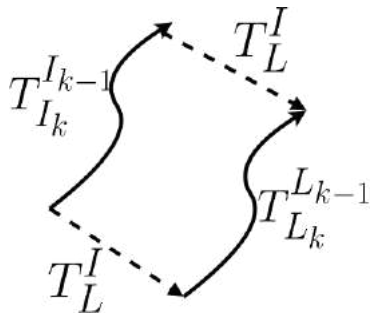


Figure 3.4: Motion based calibration constraint between LiDAR and IMU. $T_{L_k}^{L_{k-1}} \in SE(3)$ is LiDAR motion, $T_{I_k}^{I_{k-1}} \in SE(3)$ is IMU motion.

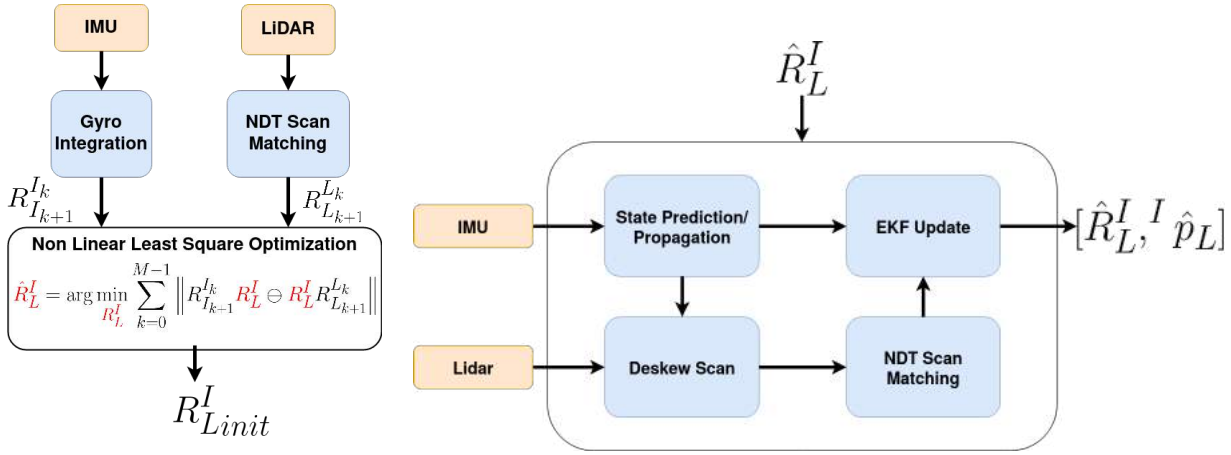
3.5 Problem Formulation

The goal is to determine the spatial 6 DoF separation, *viz.* the extrinsic calibration $T_L^I \in SE(3)$ between a 3D-LiDAR and an IMU (Figures 3.1 & 3.7). The calibration process is divided into three steps *viz* data collection (Section 3.5.1), inter sensor rotation initialization (Section 3.5.2, Figure 3.5a) and full extrinsic calibration (Section 3.5.3, Figure 3.5b).

3.5.1 Data Collection

Data collection is an important step in any sensor registration algorithm. Since the sensor suite (Figures 3.1 & 3.7) involves an IMU, a proprioceptive sensor which can sense only motion, it needs to be sufficiently motion excited [2] such that all the components (Rotation + Translation) of

[2] https://youtu.be/2IX5LVTDkLc



(a) Initialization of Rotation R^I_L

(b) Extended Kalman Filter for estimating both R^I_L and ${}^Ip_L$

Figure 3.5: Figure 3.5a presents the process of determination of an initial estimate of R^I_L, Figure 3.5b presents the Extended Kalman Filter which utilizes the initial estimate of R^I_L obtained in Figure 3.5a to estimate both R^I_L and ${}^Ip_L$ together. The EKF framework can be easily expanded to perform online sensor calibration as the time taken by the estimation pipeline (preditction & update) is negligible. The only bottlenect currently is to reduce the time taken by NDT scan matching which can be done by parallelizing the scan matching process or using a GPU for the the process or by replacing NDT with some light weight method like LOAM [2].

the extrinsic calibration parameter are completely observable. However, as remarked in [8] motion excitation along at least two degrees of rotational freedom is essential for observability. This is demonstrated in Section 3.8.6. Although [8] deals in Camera IMU calibration, the remark about observability may also be extended to the 3D-Lidar IMU calibration problem because, indirectly or directly, in both the cases the exteroceptive sensor (camera in [8] and LiDAR in this context) is used as a pose estimation sensor. The data collection XYZ trajectory is shown in Figure 3.6. For the purpose of notation, M LiDAR scans were collected in the process of collecting data.

3.5.2 Rotation Estimation

The rotation between the IMU and 3D-LiDAR is estimated by using the rotation component of the motion based calibration constraint (Equation 3.1). The rotation component is given in Equation 3.2.

$$R^{I_{k-1}}_{I_k} R^I_L = R^I_L R^{L_{k-1}}_{L_k} \tag{3.2}$$

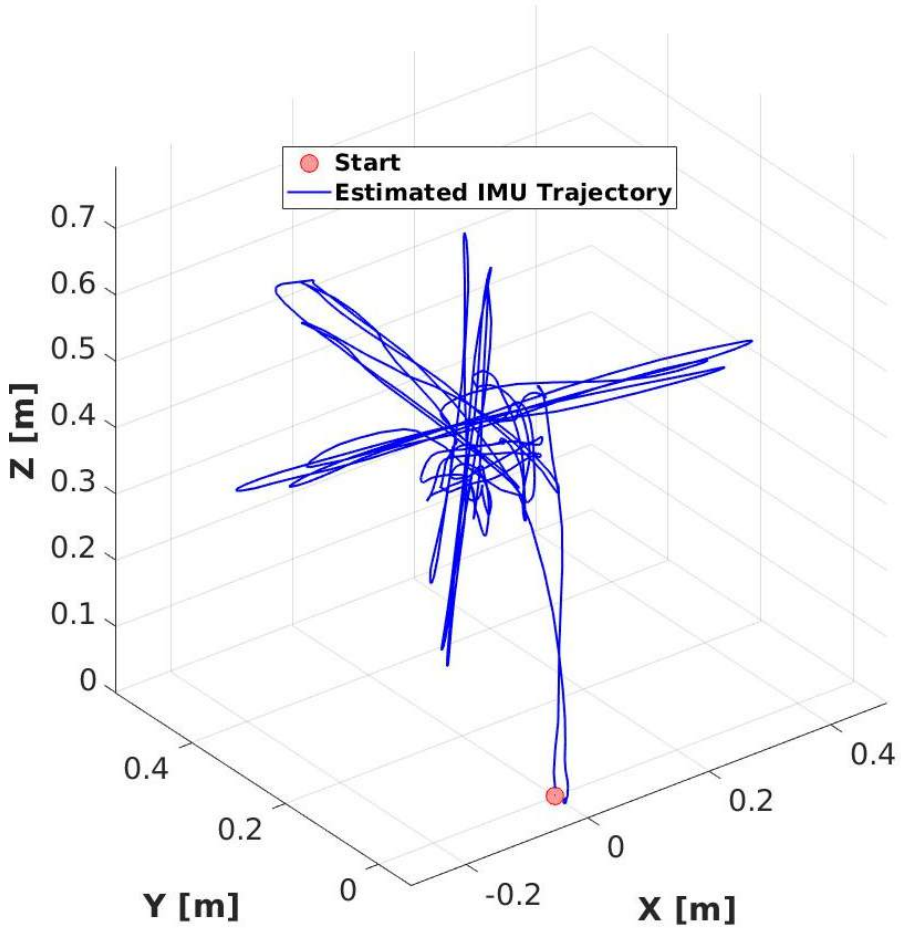


Figure 3.6: Trajectory of the LiDAR-IMU system. The framework estimates evolving IMU pose in addition to static registration between camera and LiDAR.

For rotation estimation, an axis angle representation of the sensor rotations is used. Using the axis angle representation, Equation 3.2 can be reformulated as

$$^{I_{k-1}}r_{I_k} = R_L^I \, ^{L_{k-1}}r_{L_k} \tag{3.3}$$

Here $^{I_{k-1}}r_{I_k}$ & $^{L_{k-1}}r_{L_k} \in R^3$ are axis angle representations for $R_{I_k}^{I_{k-1}}$ & $R_{L_k}^{L_{k-1}}$ respectively. As shown in Figure 3.5a, NDT scan matching [58] is used to estimate the LiDAR rotation $R_{L_k}^{L_{k-1}}$ between consecutive LiDAR scans. Gyroscope measurements between two scan instants are in-

tegrated to estimate IMU rotation $R_{I_k}^{I_{k-1}}$. An objective function (Equation 3.4), unknown in R_L^I, is formed by squaring and summing the constraint given in Equation 3.3 for each ${}^{L_{k-1}}r_{L_k}$ and the corresponding ${}^{I_{k-1}}r_{I_k}$.

$$P = \sum_{i=1}^{M-1} \left\| {}^{I_{k-1}}r_{I_k} - R_L^I \, {}^{L_{k-1}}r_{L_k} \right\|^2 \tag{3.4}$$

In order to obtain an estimate $\hat{R}_L^I$, Equation 3.4 needs to be minimized with respect to R_L^I as shown in Equation 3.5 using the Ceres [5] non linear least square solver.

$$\hat{R}_L^I = \underset{R_L^I}{\operatorname{argmin}} \, P \tag{3.5}$$

In this step, the sensor rotations $R_{I_k}^{I_{k-1}}$ and $R_{L_k}^{L_{k-1}}$ used to estimate R_L^I have certain shortcomings. First, $R_{I_k}^{I_{k-1}}$ is calculated by integrating the gyro measurements without taking the gyroscope bias into account and second, $R_{L_k}^{L_{k-1}}$ is obtained from NDT scan matching of LiDAR scans which may have significant motion distortion due to the motion undertaken by the sensor suite during the data collection step (Section 3.5.1). This step only provides an initial estimate of R_L^I which is used for initializing the EKF based algorithm described in Section 3.5.3.

3.5.3 Full State Estimation using an Extended Kalman Filter

The estimation of inter sensor translation ${}^{I}p_L$ depends on IMU translation ${}^{I_{k-1}}p_{I_k}$ (Equation 3.1) which involves double integration of IMU accelerometer measurements, but performing double integration without the knowledge of biases will introduce significant errors. An Extended Kalman Filter (EKF) is used, which, in addition to estimating R_L^I & ${}^{I}p_L$, also estimates the accelerometer & gyroscope biases, the pose & velocity of the IMU at the LiDAR scan instants. The block diagram for the EKF approach is shown in Figure 3.5b. The states estimated in the calibration process are:

$$\mathcal{X} = \{X_{I_{k=0:M-1}}^G, T_L^I\} \tag{3.6}$$

Here M is the number of scans. $X_{I_k}^G$ is the IMU state at scan timestamp k. The EKF state vector has an evolving component and a static component. The evolving component is the IMU state at scan timestamp k:

$$\hat{X}_{I_k}^G = \{{}_G^{I_k}\hat{\bar{q}}, {}^G\hat{\mathbf{v}}_{I_k}, {}^G\hat{\mathbf{p}}_{I_k}, \hat{\mathbf{b}}_{g,k}, \hat{\mathbf{b}}_{a,k}\} \tag{3.7}$$

${}_G^{I_k}\hat{\bar{q}}$ is the unit quaternion which encodes the IMU orientation such that the rotation matrix $R^\top({}_G^{I_k}\hat{\bar{q}})$ is the IMU orientation with respect to the global frame G. ${}^G\hat{\mathbf{p}}_{I_k}$ & ${}^G\hat{\mathbf{v}}_{I_k}$ are IMU position and velocity vectors ($\in R^{3\times1}$) respectively in frame G. $\hat{\mathbf{b}}_{a,k}$ & $\hat{\mathbf{b}}_{g,k}$ are the accelerometer and gyro bias vectors ($\in R^{3\times1}$) respectively. The static component of the EKF state vector is the extrinsic calibration, parameterized as T_L^I. T_L^I is formed by rotation matrix R_L^I and translation vector ${}^I\mathbf{p}_L$, however in the EKF formulation the rotation R_L^I is parameterized as a unit quaternion ${}_L^I\bar{q}$.

3.5.3.1 State Propagation

$$\begin{aligned}
{}_G^{I_{i+1}}\hat{\bar{q}} &= \exp\left(\frac{1}{2}\Omega(\boldsymbol{\omega}_{m,i} - \hat{\mathbf{b}}_{g,i})\Delta t\right){}_G^{I_i}\hat{\bar{q}} && (3.8)\\
{}^G\hat{\mathbf{v}}_{I_{i+1}} &= {}^G\hat{\mathbf{v}}_{I_i} - {}^G\mathbf{g}\Delta t + \hat{\mathbf{R}}_{I_i}^G(\mathbf{a}_{m,i} - \hat{\mathbf{b}}_{a,i})\Delta t && (3.9)\\
{}^G\hat{\mathbf{p}}_{I_{i+1}} &= {}^G\hat{\mathbf{p}}_{I_i} + {}^G\hat{\mathbf{v}}_{I_i}\Delta t - \frac{1}{2}{}^G\mathbf{g}\Delta t^2 \\
&\quad + \frac{1}{2}\hat{\mathbf{R}}_{I_i}^G(\mathbf{a}_{m,i} - \hat{\mathbf{b}}_{a,i})\Delta t^2 && (3.10)\\
\hat{\mathbf{b}}_{g,i+1} &= \hat{\mathbf{b}}_{g,i} && (3.11)\\
\hat{\mathbf{b}}_{a,i+1} &= \hat{\mathbf{b}}_{a,i} && (3.12)\\
{}_L^I\hat{\bar{q}}_{i+1} &= {}_L^I\hat{\bar{q}}_i && (3.13)\\
{}^I\hat{\mathbf{p}}_{L,i+1} &= {}^I\hat{\mathbf{p}}_{L,i} && (3.14)
\end{aligned}$$

A discrete time implementation given in [8], [57] is used to propagate the EKF state (Equations 3.8 - 3.14) from IMU timestamp i to $i+1$. The gyroscope and accelerometer measurements $\boldsymbol{\omega}_{m,i} \in R^{3\times1}$ & $\mathbf{a}_{m,i} \in R^{3\times1}$ respectively, are assumed to be constant during the IMU sampling

period Δt. In the following equations ${}^{G}\mathbf{g}$ is the acceleration due to gravity in the global frame G.

In Equation 3.8, $\exp()$ is matrix exponential (Equation 96 in [59]), $\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}]_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$ & $[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$. The gyroscope and accelerometer measurements $\boldsymbol{\omega}_{m,i}$ and $\mathbf{a}_{m,i}$ respectively, used to propagate the evolving state (Equation 3.8-3.14) state are modelled as:

$$\begin{aligned} \boldsymbol{\omega}_m &= \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\ \mathbf{a}_m &= \mathbf{a} + \mathbf{R}_G^{I\,G}\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a \end{aligned} \tag{3.15}$$

Here $\mathbf{n}_g$ & $\mathbf{n}_a$ are white Gaussian noise. Discretizing and taking expected value, Equation 3.15 can be written as:

$$\begin{aligned} \boldsymbol{\omega}_{m,i} &= \hat{\boldsymbol{\omega}}_i + \hat{\mathbf{b}}_{g,i} \\ \mathbf{a}_{m,i} &= \hat{\mathbf{a}}_i + \hat{\mathbf{R}}_G^{I_i G}\mathbf{g} + \hat{\mathbf{b}}_{a,i} \end{aligned} \tag{3.16}$$

In addition to propagation of state variables, the EKF state covariance $\mathbf{P}$ is propagated from IMU timestamp i to $i+1$ using Equation 3.17.

$$\mathbf{P}_{i+1} = \Phi(t_{i+1}, t_i)\mathbf{P}_i\Phi(t_{i+1}, t_i)^T + \mathbf{G}_i\mathbf{Q}_d\mathbf{G}_i^T \tag{3.17}$$

Here,

$$\Phi(t_{i+1}, t_i) = \begin{bmatrix} \hat{\mathbf{R}}_{I_i}^{I_{i+1}} & 0_3 & 0_3 & -\hat{\mathbf{R}}_{I_i}^{I_{i+1}} \mathbf{J}_r({}_{I_i}^{I_{i+1}}\hat{\boldsymbol{\theta}})\Delta t & 0_3 \\ -\frac{1}{2}\hat{\mathbf{R}}_{I_i}^{G}[\hat{\mathbf{a}}_i \Delta t^2]_\times & I_3 & I_3 \Delta t & 0_3 & -\frac{1}{2}\hat{\mathbf{R}}_{I_i}^{G}\Delta t^2 \\ -\hat{\mathbf{R}}_{I_i}^{G}[\hat{\mathbf{a}}_i \Delta t]_\times & 0_3 & I_3 & 0_3 & -\hat{\mathbf{R}}_{I_i}^{G}\Delta t \\ 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}$$

$$\mathbf{G}_i = \begin{bmatrix} -\hat{\mathbf{R}}_{I_i}^{I_{i+1}} \mathbf{J}_r({}_{I_i}^{I_{i+1}}\boldsymbol{\theta})\Delta t & 0_3 & 0_3 & 0_3 \\ 0_3 & -\frac{1}{2}\hat{\mathbf{R}}_{I_i}^{G}\Delta t^2 & 0_3 & 0_3 \\ 0_3 & -\hat{\mathbf{R}}_{I_i}^{G}\Delta t & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}$$

Where, $\hat{\mathbf{R}}_{I_i}^{I_{i+1}} = \exp(-\hat{\boldsymbol{\omega}}_i \Delta t)$, ${}_{I_i}^{I_{i+1}}\hat{\boldsymbol{\theta}} = -\hat{\boldsymbol{\omega}}_i \Delta t$ and $\mathbf{J}_r(\boldsymbol{\theta})$ is the right Jacobian of $SO(3)$ that maps the variation of rotation angle in the parameter vector space into variation in the tangent vector space to the manifold [60]. $\mathbf{Q}_d$ is the IMU noise covariance matrix which can be computed as done in [59] (Equations 129-130 & Equations 187-192). Computation of $\mathbf{Q}_d$ requires the knowledge of IMU intrinsic calibration parameters, *viz.* gyroscope/accelerometer noise densities and random walk (in-run biases), which can be looked up in the IMU data-sheet or determined using tools available online[3].

3.5.3.2 Deskewing Scan

The effect of removing motion distortion from LiDAR data can be seen in Figure 3.2. The 3D LiDAR sequentially produces point measurements using a rotating mechanism. When the LiDAR moves, the raw scan produced by it suffers from motion distortion. The calibration data collection process (Section 3.5.1) requires the sensor suite to exhibit motion excitation, which moves the points in a raw scan away from their true positions. In a LiDAR scan, each 3D point is measured

[3] `https://github.com/rpng/kalibr_allan`

from a temporally unique frame and comes with a timestamp (which is somewhere between two adjacent scan timestamps). In order to address the problem of motion distortion, the IMU pose at point timestamp needs to be estimated/predicted/interpolated. The IMU propagation model (Equation 3.8 - 3.14) is used for IMU pose prediction at lidar point timestamp (firing timestamp). Once an estimate of the IMU pose at point timestamp is obtained, the best known estimate of the extrinsic calibration parameter T_L^I is utilized to infer the corresponding LiDAR pose, which can be done by exploiting the motion constraint given in Equation 3.1. For example, considering a point $x_{k_i}^L$ in the k^{th} scan bearing timestamp k_i. In order to deskew this point, the Equation 3.1 is manipulated and used to estimate LiDAR motion $T_{L_{k_i}}^{L_k}$ between the scan timestamp k and the point timestamp k_i (Equation 3.18)

$$T_{L_{k_i}}^{L_k} = (\hat{T}_L^I)^{-1}(\hat{T}_{I_k}^G)^{-1}\hat{T}_{I_{k_i}}^G\hat{T}_L^I \tag{3.18}$$

$T_{L_{k_i}}^{L_k}$ calculated in Equation 3.18 is used to transform the point $x_{k_i}^L$ for obtaining a deskewed LiDAR scan. Here, $\hat{T}_L^I$ is the best known estimate of extrinsic calibration T_L^I at that instant, $\hat{T}_{I_k}^G$ is an estimate of IMU pose at scan timestamp k and finally $\hat{T}_{I_{k_i}}^G$ is the pose of the IMU at point timestamp k_i, obtained using the IMU state propagation model (Equation 3.8 - 3.14)[4].

3.5.3.3 NDT Scan Matching

After deskewing the scan, NDT scan matching [58] is used to generate LiDAR motion estimates $T_{L_k}^{L_{k-1}}$ between consecutive deskewed LiDAR scans $k-1$ and k. The scan matching algorithm is also used to make a map of the environment and localize the LiDAR in the map. So, this module gives not only incremental LiDAR motion $T_{L_k}^{L_{k-1}}$ but also LiDAR pose in a global map $T_{L_k}^{L_1}$. These LiDAR motion estimates are used as measurement in during the EKF state update step.

3.5.3.4 State Update

The State Update module requires the knowledge of a measurement model, measurement residual and the measurement Jacobians with respect to the state variables. This section presents the

[4]Check the effect of deskewing in the video posted here `https://youtu.be/YmTyoQA4NaY`

measurement model and the measurement residual, but the derivation of measurement Jacobians has been omitted here the interest of space (however more details about calculating Jacobians can be found in Chapter A, Sections A.1 & A.2). As described in Section 3.5.3.3, the result of NDT scan matching, parameterized as $T_{L_k}^{L_{k-1}} \in SE(3)$, is used as measurement. The *motion based calibration* constraint in Equation 3.1 has been used to derive the measurement model. Manipulating Equation 3.1 gives the measurement model as presented in Equation 3.19:

$$T_{L_k}^{L_{k-1}} = (T_L^I)^{-1}(T_{I_{k-1}}^G)^{-1}T_{I_k}^G T_L^I \tag{3.19}$$

The LHS of Equation 3.19 is the measurement and the RHS is a function of state variables T_L^I, $T_{I_{k-1}}^G$, $T_{I_k}^G$. So, the measurement model (Equation 3.19) is in agreement with the standard form $z = h(x)$ used in EKF, where z is the measurement and $h()$ is the measurement model which is a function of the state x. In the current scenario, measurement $z = T_{L_k}^{L_{k-1}}$ & measurement model $h(x) = (T_L^I)^{-1}(T_{I_{k-1}}^G)^{-1}T_{I_k}^G T_L^I$ and state $x = \{T_{I_{k-1}}^G, T_{I_k}^G, T_L^I\}$. Here,

$$T_L^I = \begin{bmatrix} \mathbf{R}({}_L^I\bar{q}) & {}^I\mathbf{p}_L \\ 0 & 1 \end{bmatrix}, T_{I_{k-1}}^G = \begin{bmatrix} \mathbf{R}^T({}_G^{I_{k-1}}\bar{q}) & {}^G\mathbf{p}_{I_{k-1}} \\ 0 & 1 \end{bmatrix}$$

$$T_{I_k}^G = \begin{bmatrix} \mathbf{R}^T({}_G^{I_k}\bar{q}) & {}^G\mathbf{p}_{I_k} \\ 0 & 1 \end{bmatrix}$$

Clearly T_L^I, $T_{I_{k-1}}^G$, $T_{I_k}^G$ depend on state variables.

Separating the rotation and translation components of Equation 3.19, one obtains.

$$
\begin{aligned}
R_{L_k}^{L_{k-1}} &= \mathbf{R}^{\top}({}_{L}^{I}\bar{q})\mathbf{R}({}_{G}^{I_{k-1}}\bar{q})\mathbf{R}^{T}({}_{G}^{I_k}\bar{q})\mathbf{R}({}_{L}^{I}\bar{q}) \\
{}^{L_{k-1}}p_{L_k} &= \mathbf{R}^{\top}({}_{L}^{I}\bar{q})\Bigg[\Bigg(\mathbf{R}({}_{G}^{I_{k-1}}\bar{q})\mathbf{R}^{T}({}_{G}^{I_k}\bar{q}) - \mathbf{I}\Bigg){}^{I}\mathbf{p}_{L} \\
&\quad + \mathbf{R}({}_{G}^{I_{k-1}}\bar{q})\Bigg({}^{G}\mathbf{p}_{I_k} - {}^{G}\mathbf{p}_{I_{k-1}}\Bigg)\Bigg]
\end{aligned} \tag{3.20}
$$

The measurement model calculated at state estimates gives the predicted rotation and translation measurement, *viz.* $\hat{R}_{L_k}^{L_{k-1}}$ and ${}^{L_{k-1}}\hat{p}_{L_k}$ respectively. The difference between the true measurements and predicted measurements gives the measurement residual $\mathbf{r}_k$ required for state update (Equation 3.21).

$$
\mathbf{r}_k = \begin{bmatrix} \texttt{Log}(R_{L_k}^{L_{k-1}}(\hat{R}_{L_k}^{L_{k-1}})^{\top}) \\ {}^{L_{k-1}}p_{L_k} - {}^{L_{k-1}}\hat{p}_{L_k} \end{bmatrix} \tag{3.21}
$$

Here, `Log()` associates a matrix $R \in SO(3)$ to a vector $\in R^{3\times 1}$ (via a skew symmetric matrix). In addition to the measurement residual $\mathbf{r}_k$, one also requires the Jacobians of the measurement model with respect to the state variables in order to perform state and covariance update. The Jacobians (Equation 3.22) are evaluated at the best available estimate of the state variables $x = \{T_{I_{k-1}}^{G}, T_{I_k}^{G}, T_{L}^{I}\}$.

$$
\begin{aligned}
\mathbf{H}_{T_L^I}^{T_{L_k}^{L_{k-1}}} &= \left.\frac{\partial T_{L_k}^{L_{k-1}}}{\partial T_L^I}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^{G},\hat{T}_{I_k}^{G},\hat{T}_{L}^{I}\}} \\
\mathbf{H}_{T_{I_{k-1}}^{G}}^{T_{L_k}^{L_{k-1}}} &= \left.\frac{\partial T_{L_k}^{L_{k-1}}}{\partial T_{I_{k-1}}^{G}}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^{G},\hat{T}_{I_k}^{G},\hat{T}_{L}^{I}\}} \\
\mathbf{H}_{T_{I_k}^{G}}^{T_{L_k}^{L_{k-1}}} &= \left.\frac{\partial T_{L_k}^{L_{k-1}}}{\partial T_{I_k}^{G}}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^{G},\hat{T}_{I_k}^{G},\hat{T}_{L}^{I}\}}
\end{aligned} \tag{3.22}
$$

These individual Jacobians are stacked together to form a consolidated Jacobian $\mathbf{H}_k$ and used for

state update when a measurement update is available. The state update equations are presented in Equations 3.23-3.25.

$$\mathbf{K}_k = \mathbf{P}_{k\_}\mathbf{H}_k^\top(\mathbf{H}_k\mathbf{P}_{k\_}\mathbf{H}_k^\top + \mathbf{R})^{-1} \tag{3.23}$$

$$\begin{bmatrix} \hat{X}_{I_{k+}}^G \\ \hat{T}_{L+}^I \end{bmatrix} = \begin{bmatrix} \hat{X}_{I_{k-}}^G \\ \hat{T}_{L-}^I \end{bmatrix} \oplus \mathbf{K}_k\mathbf{r}_k \tag{3.24}$$

$$\mathbf{P}_{k_+} = \mathbf{P}_{k_-} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k_-} \tag{3.25}$$

Here $\mathbf{K}_k$ is the Kalman gain which is used in Equations 3.24 and 3.25 for state and state covariance update respectively. $\mathbf{R}$ is the tunable measurement covariance matrix. '-' denotes the estimate before update while '+' denotes the estimate after update. $\oplus$ in Equation 3.24 refers to generic composition which can be algebraic addition for variables on vector space or rotation composition for variables on $SO(3)$.

3.6 System Description

The sensor system (Figures 3.1 & 3.7) used in this chapter consists of a Ouster 128 Channel LiDAR and a Vectornav VN-300 IMU. The LiDAR outputs scans at 10 Hz and IMU outputs gyroscope and accelerometer measurements at 400 Hz.

3.7 Experiments and Results

The utility of the proposed registration algorithm is demonstrated by calibrating the sensor suite shown in Figure 3.7. As the system consists of an Ouster 128 channel LiDAR, in addition to using it in 128 channel mode, one can reconfigure it to 16, 32, 64 channel modes as well. Therefore, experiments have been done to present the performance of the proposed algorithm when the LiDAR is used with lower number of channels/rings, with an aim to establish that the proposed algorithm gives comparative performance irrespective of the number of channels in the LiDAR. Two different datasets 1 & 2 have been collected by placing the IMU at two different locations *viz.* locations 1 & 2 (see Figure 3.7) respectively. In the following sections, all the Figures (plots)

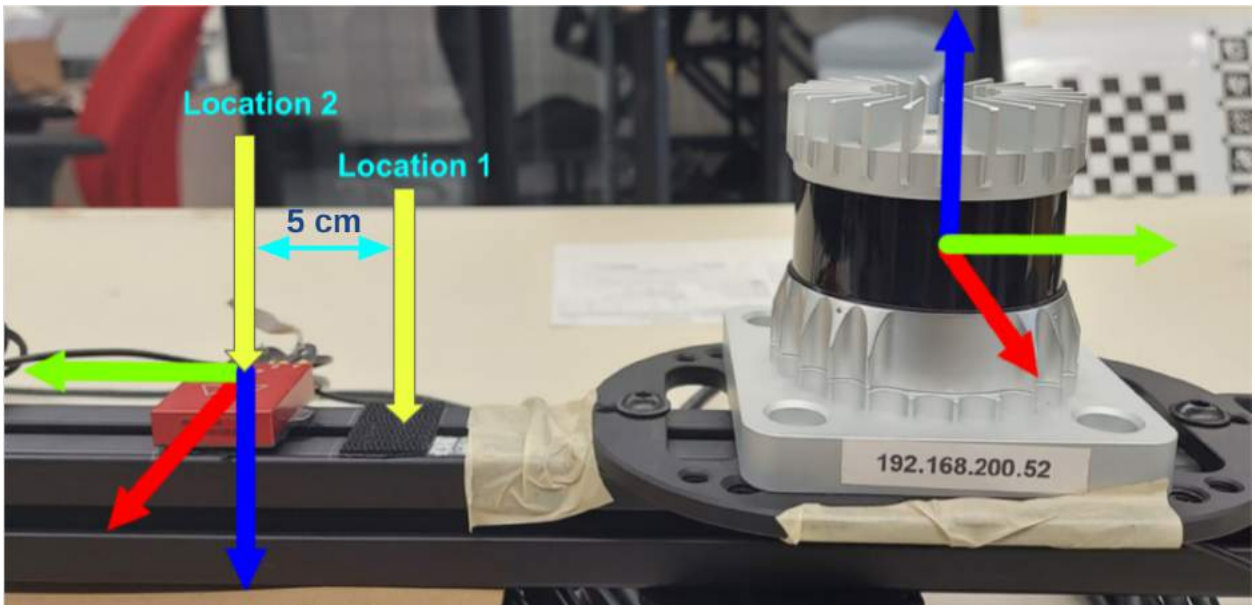


Figure 3.7: **Experimental Platform:** LiDAR Inertial Sensor Suite with an Ouster 128 Channel LiDAR and a Vectornav-VN 300 IMU. Red: x axis, Green: y-axis, Blue: z-axis. The IMU is placed at two different locations (highlighted by yellow arrows), which, according to ruler measurements, are apart from each other by 5 cm along the y axis. In the absence of ground truth, this information shall be used as a reference to validate the proposed registration/calibration algorithm.

have been generated using dataset 2 with the 3D-LiDAR operating in 128 channel mode. Similar results/plots were obtained with all the channel modes, and for both the datasets, but those have been omitted here in the interest of space.

3.8 Discussion

3.8.1 Convergence

Figures 3.8 and 3.9 show the convergence of the estimated extrinsic calibration (rotation and translation respectively). As far as the translation variables are considered, the filter is initialized at ${}^{I}\hat{p}_{L} = [0, 0, 0]$, and the filter converges to fixed values with a tight $\pm 1\sigma$ bound. As far as rotation variables are considered the filter has been initialized with estimates obtained from the rotation initialization technique from Section 3.5.2. The convergence of the rotation variables with the EKF is shown in Figure 3.9. Although it is difficult to check the accuracy of the converged result without ground truth data, the results obtained is close to the ruler measurements.

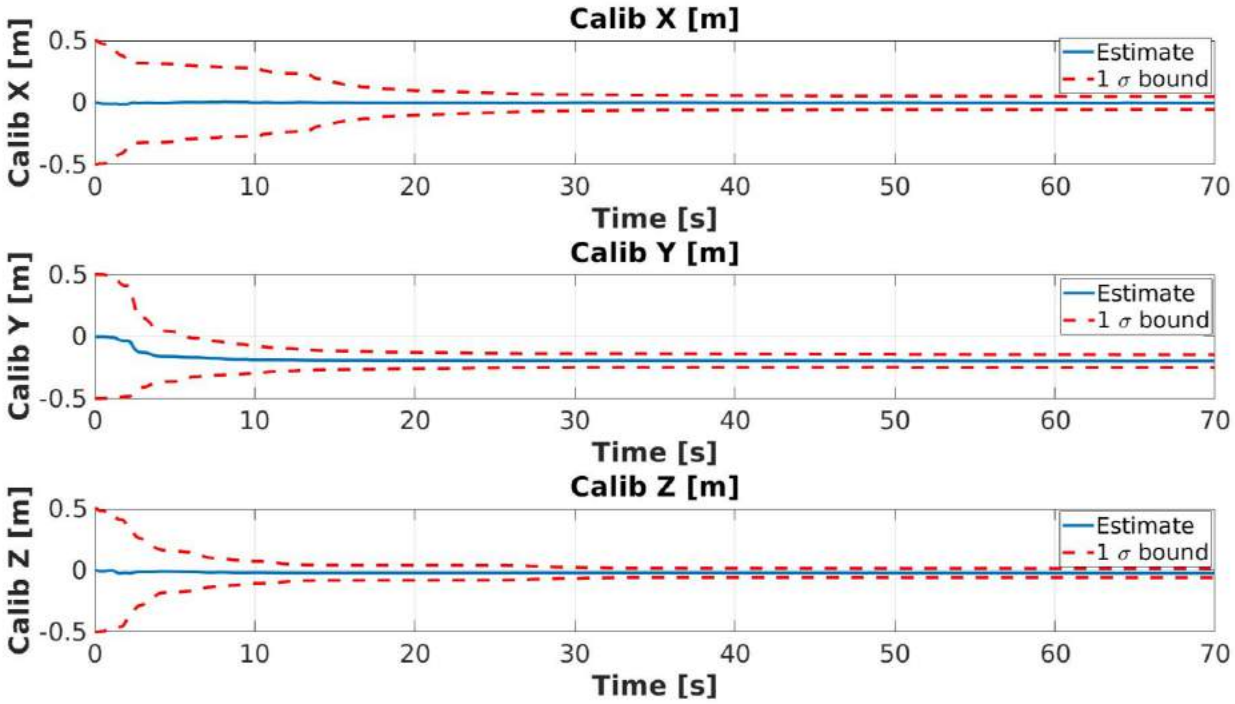


Figure 3.8: Calibration Results for translation component ${}^{I}p_L$ and 1 σ bounds. The filter starts from ${}^{I}\hat{p}_L = [0, 0, 0]$ and converges to a fixed value.

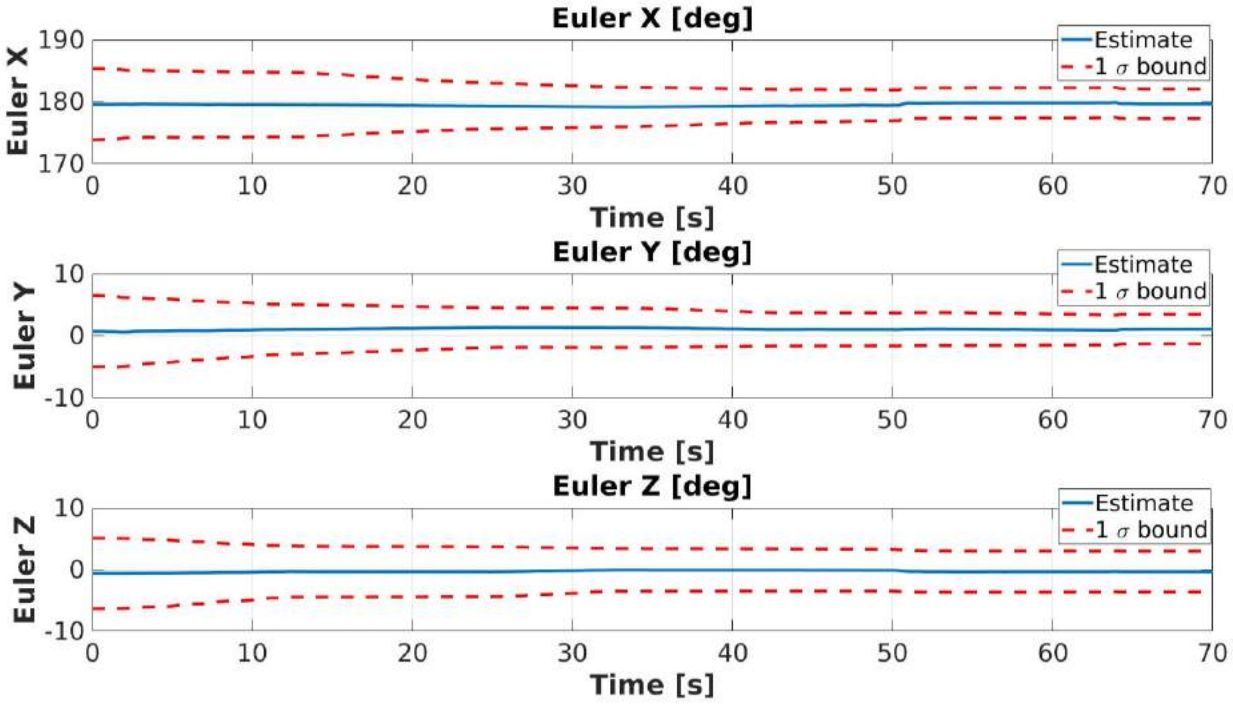


Figure 3.9: Calibration Results for rotation component and 1 σ bounds. The filter starts from initialization values that initial rotation component (Section 3.5.2) estimation reports.

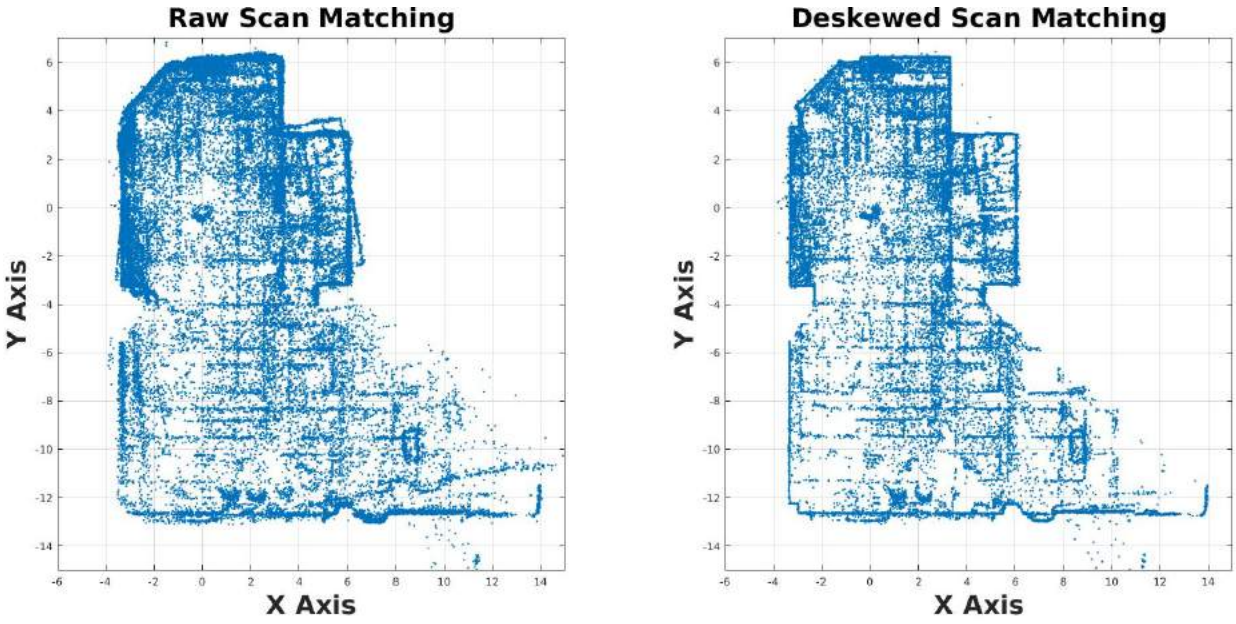


Figure 3.10: Scan Matching with raw and deskewed scans during the calibration process. The scans have been downsampled to improve visibility.

3.8.2 Importance of deskewing LiDAR scans

This section discusses the importance of deskewing the LiDAR scans (Section 3.5.3.2) during the EKF based calibration process (Section 3.5.3). Figure 3.10 presents the result of scan matching obtained during the calibration process without and with deskewing of LiDAR scans. The left image in Figure 3.10 shows blurred and misaligned edges/corners. The edges/corners are blurred because the scan is not deskewed and they are misaligned because the LiDAR Odometry from raw scan matching is not correct. Motion distorted LiDAR scans lead to inferior results of LiDAR Odometry which in turn results in overall deterioration of the calibration estimates (see Figure 3.11). Whereas, in the right image of Figure 3.10 the edges/corners are sharp and not blurred, thanks to the deskewing of LiDAR scans, done using the best available estimate of the extrinsic calibration $\hat{T}_L^I$ during the EKF based calibration process. The evolution of calibration results is presented in Figure 3.11. The results without deskewing do not converge to fixed values and vary with time when compared against the results obtained when the LiDAR scans are deskewed.

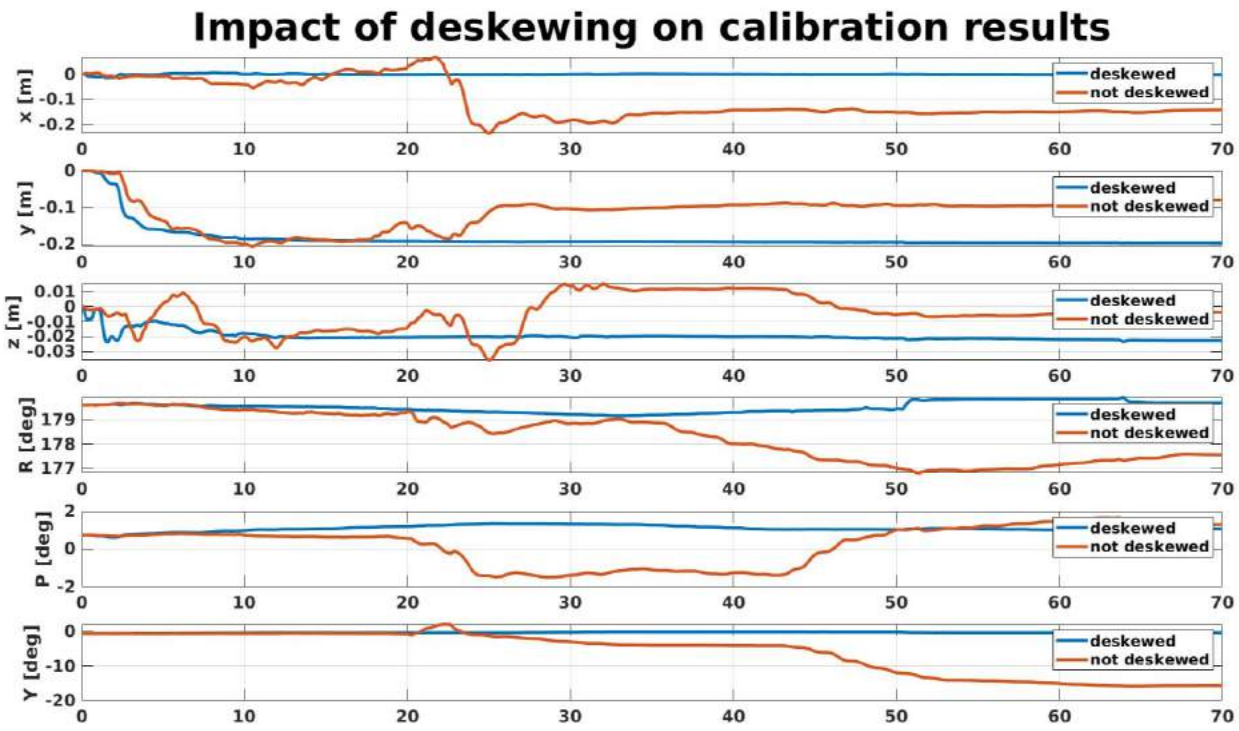


Figure 3.11: The calibration results do not converge to fixed values when the LiDAR scan is not deskewed.

3.8.3 Changing LiDAR Density

Tables 3.1 and 3.2 present the final calibration estimates obtained at the end of the Kalman Filtering process, for different ring numbers (i.e. different channel modes), with datasets 1 & 2 collected at locations 1 & 2 (see Figure 3.7) respectively. In both Tables 3.1 & 3.2 it's found that the calibration algorithm converges to comparable estimates of calibration parameters irrespective of the number of rings/channels, thus proving its usability across LiDARs with varying channel numbers.

3.8.4 Changing IMU Frequency

The effect of changing IMU frequency on the calibration result has been presented in Table 3.3 where it can be seen that irrespective of the IMU frequency the change in calibration estimates is not substantial, thus proving that the proposed technique will work not only for different LiDAR point densities but also with different IMU frequencies.

No. rings	**R**°	**P**°	**Y**°	**x** [m]	**y** [m]	**z** [m]
128	-180.0355	-0.0956	-0.7801	0.0044	-0.1455	-0.0246
64	-179.9378	-0.1337	-0.8293	0.0028	-0.1458	-0.0234
32	-179.9213	-0.0311	-1.0018	0.001	-0.1448	-0.0210
16	-179.9129	0.1576	-1.0825	-0.0035	-0.1458	-0.0235

Table 3.1: Results with IMU at Location 1 with the LiDAR used with different channel modes

No. rings	**R**°	**P**°	**Y**°	**x** [m]	**y** [m]	**z** [m]
128	-179.6770	1.0774	-0.3735	-0.0024	-0.1971	-0.0227
64	-179.1621	0.8255	-0.0392	-0.0020	-0.1956	-0.0248
32	-179.2619	0.8368	-0.2595	-0.0031	-0.1976	-0.0231
16	-179.1540	1.1280	-0.3603	-0.0041	-0.2005	-0.0241

Table 3.2: Results with IMU at Location 2 with the LiDAR used with different channel modes

3.8.5 Relative Verification

The usual ways to validate calibration algorithms is to use simulators, auxiliary sensors like cameras or GPS, or, in the case of OEMs, CAD diagrams as ground truth. As the sensor suite (Figure 2.3b) has been assembled by procuring sensors from different vendors, there is no reference to compare against. Furthermore, using any auxiliary sensor also involves an additional overhead

IMU-Freq	x[cm]	y[cm]	z[cm]	r_x°	r_y°	r_z°
400 Hz	0.3459	-19.8857	-2.5021	-179.5628	1.6001	-2.5267
200 Hz	0.4552	-20.3478	-2.8486	179.8637	0.7028	-1.4544
100 Hz	0.3483	-19.7730	-2.1628	179.7881	0.6604	-2.5549
50 Hz	1.6108	-20.1753	-2.1024	179.3356	1.9790	-4.9149

Table 3.3: **Effect of changing IMU frequency on calibration estimates**. xyz in cm, euler angles in degree.

No. rings	R°	P°	Y°	x [m]	y [m]	z [m]
128	0.3424	1.1778	0.4061	-0.0076	0.0516	0.0019
64	0.5886	0.9678	0.7887	-0.0056	0.0497	0.0014
32	0.6442	0.8793	0.7420	-0.0050	0.0527	0.0021
16	0.7404	0.9846	0.7244	-0.0016	0.0547	0.0005

Table 3.4: Difference between calibration results for datasets collected at Location 1 and Location 2 with varying ring density. The difference along y-axis has been highlighted in blue.

of calibrating that sensor w.r.t the IMU. In this work a relative verification technique is adopted that can validate calibration results against a reference obtained by ruler measurement. As described earlier, two datasets 1 & 2 were collected with the IMU kept at two different locations 1 & 2 respectively (see Figure 3.7). The relative offset between the two IMU locations is 5 cm along y-axis according to ruler measurement and this has been used as a reference to compare against the results from the proposed calibration technique. In order to validate the proposed calibration algorithm, it's run on datasets collected at both of these locations and for all channel modes, and the difference in the calibration for both the locations has been presented in Table 3.4 (which is the difference between the results presented in Tables 3.1 and 3.2). Using the aforementioned ruler measurement as reference, percentage errors of 3.2 %, 0.6 %, 5.4 % & 9.4% were achieved along the y-axis for 128, 64, 32 & 16 channel modes respectively.

3.8.6 Motion Required for Observability of Extrinsic Calibration

As mentioned in [8], it is advisable to excite at least two degrees of rotational freedom for sufficiently long time at the beginning of the calibration process in order to ensure convergence to reliable calibration values. This is demonstrated on comparing Figures 3.8 & 3.9 which show the convergence of the calibration parameters, against Figures 3.12 & 3.13 which present the rotation and translation trajectory undertaken by the sensor suite during the data collection procedure (Section 3.5.1). In Figure 3.8, the translation calibration parameters converge in 25-30 seconds which correspond to the time when the rotation motions (see Figure 3.12) about two axes (*viz.* X & Z) cease to be performed. As far as the rotation calibration parameters are considered (Figure 3.9)

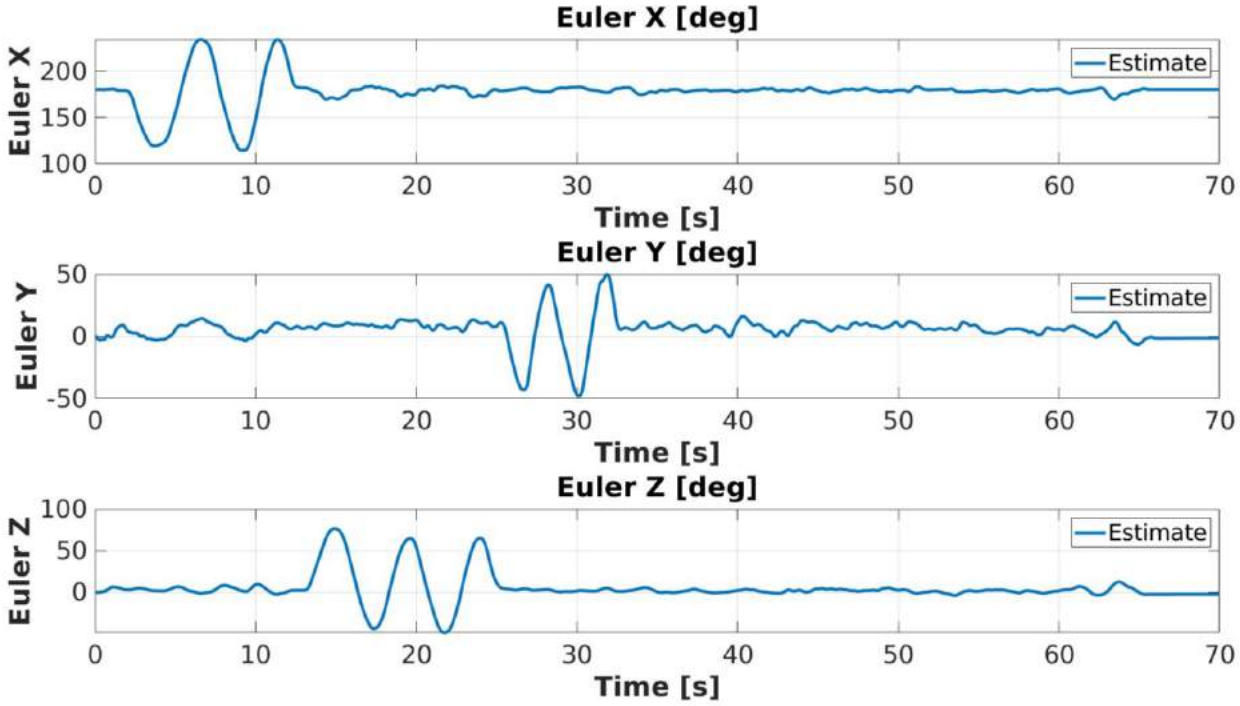


Figure 3.12: Rotation trajectory during data collection

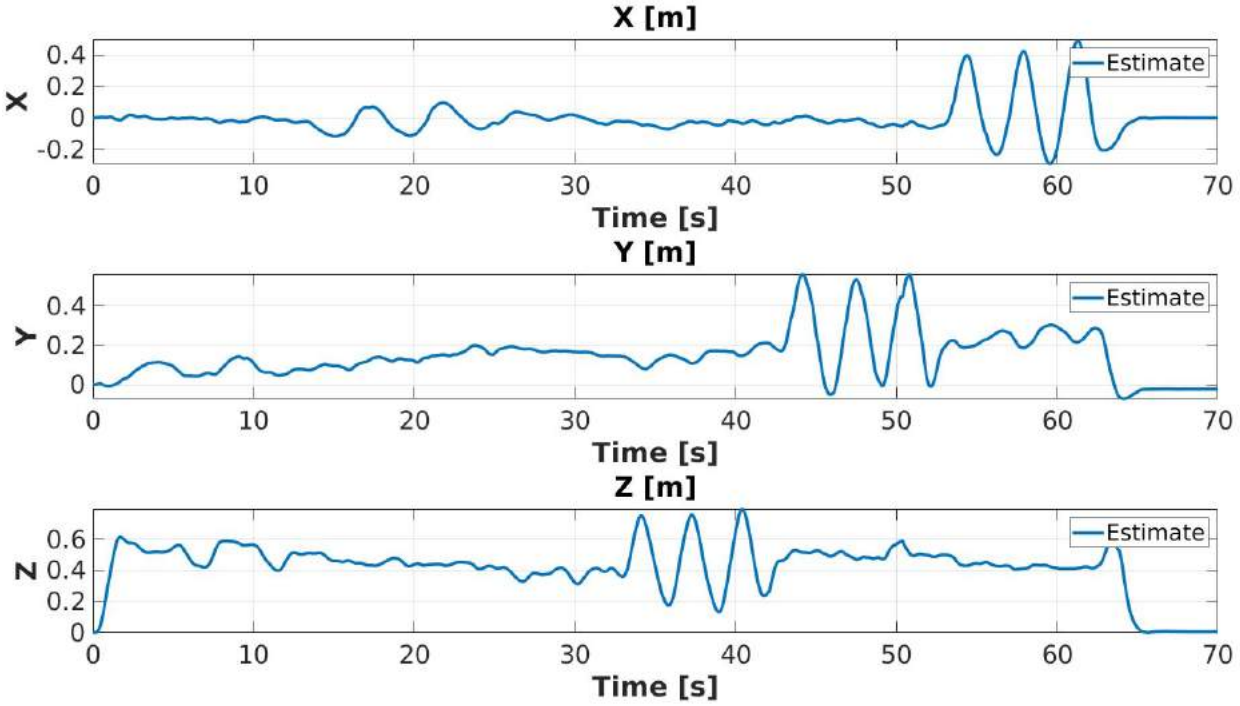


Figure 3.13: XYZ trajectory during data collection

they vary marginally during the EKF based calibration process (Section 3.5.3) as they are already initialized close to final estimates during the initialization procedure (Section 3.5.2).

3.9 Advantages of using an EKF framework

Firstly, the filtering framework can be easily extended to online applications as the time taken by the estimation modules (prediction and update) is negligible when compared against time taken by a comparable batch estimation framework given in [1] and [16]. In the proposed framework (Figure 3.5b) the NDT scan matching module takes the maximum time to execute but this process can be easily made faster by parallelizing the operation or by accelerating it by using GPUs if necessary or by using a faster LiDAR odometry algorithm like LOAM [2]. As the framework also estimates evolving IMU pose and velocity it is basically akin to a SLAM framework with added benefits of performing extrinsic calibration. So, when operating it as a SLAM module one may use LOAM which is faster and can operate in real time and while performing extrinsic calibration offline one may use NDT scan matching. Secondly, the proposed EKF framework can easily accomodate new residuals like the point to plane constraints used in [1] and [16] in the EKF update step on top of the motion based update (Section 3.4 & 3.5.3.4) already discussed in the chapter, thus providing additional ways to constrain the variables being estimated. Thirdly, the EKF framework can easily accomodate multi-rate sensors because of its ability to perform an update whenever a measurement arrives and perfom predictions or propagation until then. Fourth, the EKF approach, because of the ease with which it handles different kind of residuals and multi-rate measurements, can easily accomodate different sensing modalities thus helping with registration of multiple sensors together.

3.10 Conclusion

This Chapter proposes an EKF based 3D-LiDAR IMU calibration algorithm which does not depend on any calibration target or auxiliary/aiding sensors for estimating the extrinsic calibration. This method can be used for LiDARs of varying pointcloud density (Section 3.8.3) and different IMU frequencies (Section 3.8.4). Moreover, it requires no manual or tape measured intialization.

The Chapter experimentally presented the type of motion required for observing the calibration estimates (Section 3.8.6). The work also presented a relative way of validating calibration algorithms when there's no access to ground truth or an auxiliary sensor at one's disposal. Although the relative verification technique tells us how well the calibration algorithm works, it's impractical in a real scenario where the IMUs and LiDARs would be rigidly attached and it's not easy to move a sensor by a known amount to check the correctness of the results. Moreover, relative verification cannot help us detect systematic errors in estimation which may be present as a constant offset in the estimated pose but will go undetected under relative verification.

In a broader context, the EKF based filtering technique not only estimates the static registration parameters between rigidly attached sensors but also the evolving/changing IMU pose (Figures 3.12 & 3.13) and velocity in addition to building a map of the environment as shown in Figure 3.10. This implies that the estimation process is akin to a SLAM algorithm with the additional advantage of performing extrinsic calibration of the sensor components used in the SLAM process thus paving the path for doing multi-sensor extrinsic calibration, localization and mapping simultaneously.

The thesis started with a proposed technique to perform Camera-LiDAR registration using a calibration target in Chapter 2. In the current Chapter 3, a novel method to perform targetless extrinsic calibration of LiDAR-IMU system was introduced which shows promising results not only for static estimating extrinsic calibration but also for performing multi-sensor localization and mapping simultaneously. In the next Chapter 4, the thesis presents two different alternatives to perform Camera-IMU registration under an EKF approach. The aim is to judge which among the presented Camera IMU registration approaches perform better and to use the same as a building block in the ultimate goal to develop LiDAR-IMU-camera joint extrinsic calibration in Chapter 5.

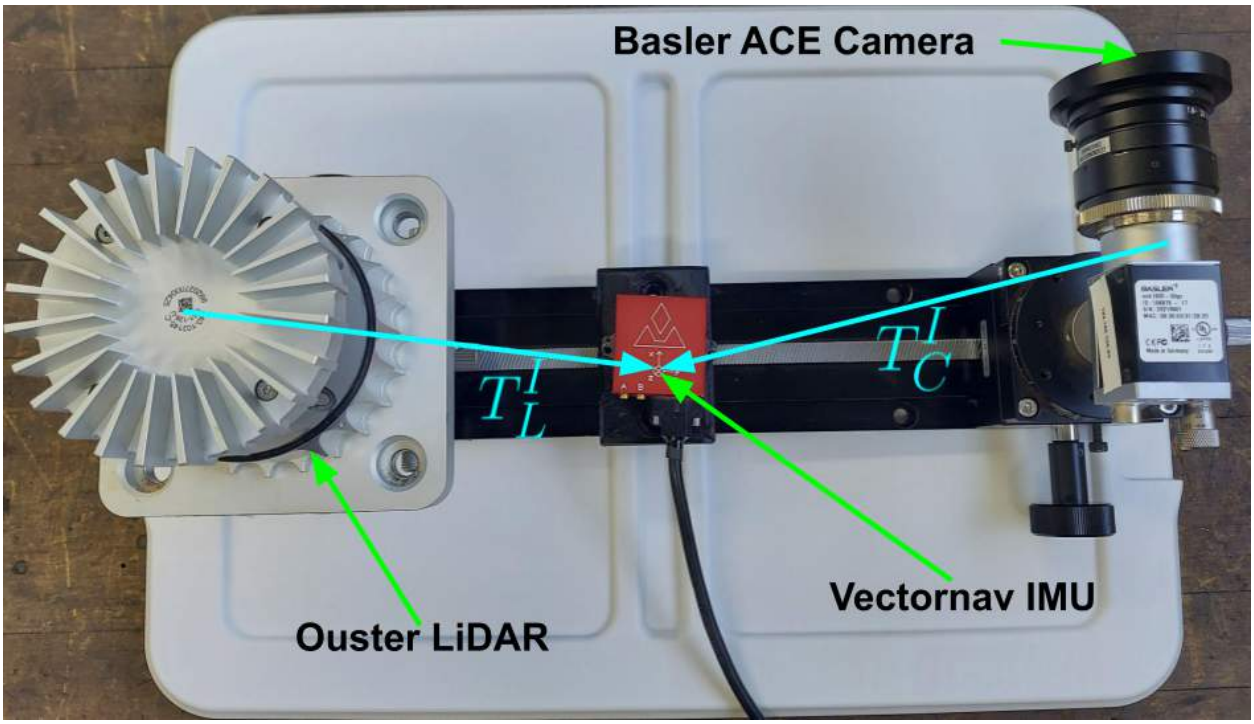


Figure 4.1: **IMU Camera System:** Consists a Basler Ace Camera $[1600 \times 1200]$ running at 30 Hz, and a Vectornav VN-300 Camera running at 400 Hz. The goal of this chapter is estimate camera-IMU extrinsic calibration T_C^I.

4.1 Introduction

Camera-IMU calibration is an important topic in robotics as it is a necessary prerequisite for Visual Inertial Odometry (VIO) and SLAM (e.g. [57], [61]). Well known works on camera-IMU cross calibration are the filtering approach provided in [8], and smoothing approaches presented in [7] and [62]. All of the above techniques use a checkerboard/fiducial marker in order to obtain camera measurement updates for their respective measurement residuals. While [8] and [7] use the fiducial marker's corner detection as measurement residuals, [62] uses the camera as a pose sensor. Among these methods [7] is more popular because the authors have open-sourced their

work (`Kalibr-toolbox`) and currently it happens to be the most widely used camera-IMU calibration algorithm in the robotics community.

4.2 Literature Review

Camera IMU calibration is a much studied problem in robotics. Broadly, available literature can be divided into methods that use filtering approach, and those that use smoothing (batch optmization) approach. The common characteristic of all the methods is that virtually all of them depend on a calibration target or artifical landmarks. [8] is probably the first Camera IMU calibration algorithm for the robotics community. This method uses an Error State Extended Kalman Filter (EKF) for camera-IMU spatio-temporal cross calibration. The method requires a good initialization which comes from tape measurements. The authors of [8] also explore the observability of the Kalman Filter with the conclusion that only rotations in two degrees of freedom are necessary to estimate the camera IMU cross calibration. Another filtering approach, similar to [8] but using an Unscented Kalman Filter (UKF) instead of EKF is presented in [63]. Among smoothing approaches [7] and [62] are prominent. Both the methods are similar in the sense that they model the trajectory of the camera inertial sensor suite as a continuous time B-spline, however they differ in the way they model the camera measurement. In [62] the camera is used as a pose sensor where the pose is estimated by detecting a checkerboard of known dimensions, whereas in [7] the camera uses detection of the checkerboard (of known dimensions) corners as the measurement model. [7] is more popular because its authors have also released an open source implementation[1] thus making it the most widely used and currently the only open sourced camera IMU calibration algorithm in the robotics community.

4.3 Contribution

The contribution of this chapter is that the author presents his own implementation of [8] and performs two different comparisons by means of extensive experimental evaluation. First, the results from this self implementation of [8] is compared against the case when the camera is used

[1] `https://github.com/ethz-asl/kalibr/wiki`

as a pose sensor (as done for the LiDAR sensor in Chapter 3) using the *motion based calibration* constraint ([18], [42]) explained in Section 3.4. In the context of camera-IMU calibration this is much like the smoothing approach [62], albeit under a filtering framework. Second, the results of the self implementation of [8] is compared against the famous `Kalibr-toolbox` [7]. The author has also open-sourced a working implementation [2] of the developed code base for the robotics community, because at the moment there is no known alternative to `Kalibr-toolbox` which is becoming increasingly tedious to install and use under newer Ubuntu and Python distributions. By the end of this Chapter, one would be able to pick whether the use of camera as a pose sensor is better or if the use of camera as sensor which obtains pixel measurements of known 3D positions on a calibration target is more suited for performing extrinsic calibration. Similar to LiDAR-IMU extrinsic calibration method in Chapter 3, the present chapter also estimates evolving IMU pose and velocity in addition to estimating the static extrinsic calibration between the camera and the IMU. Finally, the implementation presented in this Chapter paves the path for joint calibration of LiDAR-IMU-camera system in Chapter 5.

4.4 Motion Based Extrinsic Calibration

The motion based constraint for corresponding sensor motion is given in Equation 3.1 and a schematic is shown in Figure 4.2. This has been explained in detail in Section 3.4. In the current scenario, $T_{C_k}^{C_{k-1}}$ is the motion between two camera images which can be obtained by detecting & tracking a checkerboard using OpenCV[49], and $T_{I_k}^{I_{k-1}}$ is the motion experienced by the IMU between the two image instants. For the sensors spatially separated from each other by a fixed rigid pose T_C^I (the extrinsic calibration parameter), the motion based calibration constraint is given in Equation 4.1.

$$T_{I_k}^{I_{k-1}} T_C^I = T_C^I T_{C_k}^{C_{k-1}} \tag{4.1}$$

[2] `https://github.com/unmannedlab/camera_imu_calibration`

Here, $T_{I_k}^{I_{k-1}} = \begin{bmatrix} R_{I_k}^{I_{k-1}} & {}^{I_{k-1}}p_{I_k} \\ 0 & 1 \end{bmatrix}$, $T_{C_k}^{C_{k-1}} = \begin{bmatrix} R_{C_k}^{C_{k-1}} & {}^{C_{k-1}}p_{C_k} \\ 0 & 1 \end{bmatrix}$ and $T_L^I = \begin{bmatrix} R_L^I & {}^{I}p_L \\ 0 & 1 \end{bmatrix}$. $R_b^a \in SO(3)$ is a rotation matrix and ${}^{a}p_b \in R^{3\times 1}$ is a translation vector. The motion based constraint given by Equation 4.1 is used for estimating the inter sensor rotation $R_C^I \in SO(3)$ and translation ${}^{I}p_C \in R^3$. Specifically, the rotation component of Equation 4.1 is utilized to initialize the inter sensor rotation and use this initialization to estimate the inter sensor translation using an Extended Kalman Filter based algorithm [8], [57].

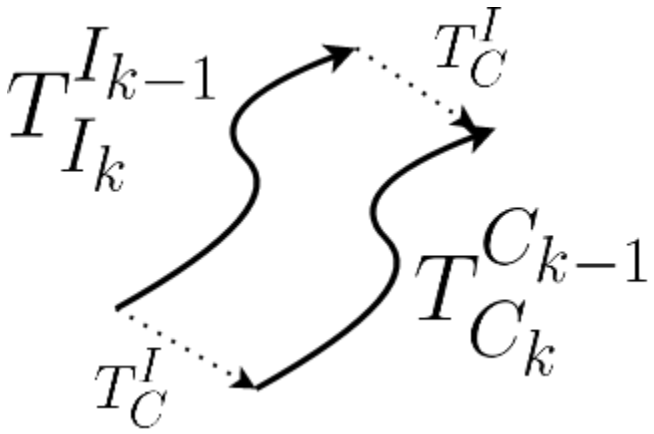


Figure 4.2: Motion based calibration constraint between Camera and IMU. Here $T_{I_k}^{I_{k-1}}$ is IMU motion, and $T_{C_k}^{C_{k-1}}$ is camera motion.

4.5 Re-projection Based Extrinsic Calibration

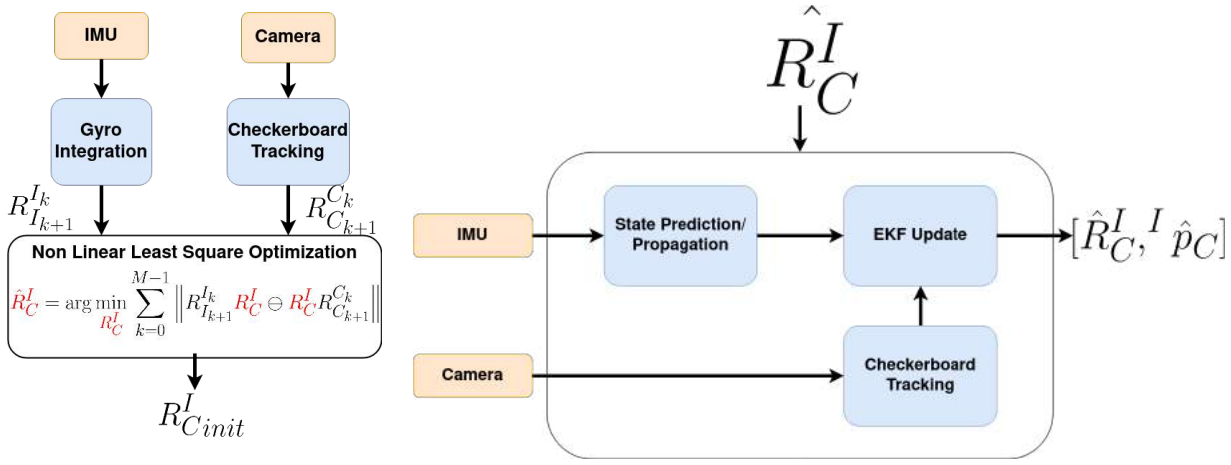
This approach has been adopted in [8] and [7] to form measurement residuals for camera-IMU extrinsic calibration. The states to be estimated e.g. camera-IMU extrinsic calibration $T_C^I \in SE(3)$ & IMU pose $T_{I_k}^G \in SE(3)$ at time k w.r.t a global frame G are used to form measurment residuals. The measurements z are the detected corners of the checkerboard pattern, which are detected using OpenCV [49].

$$z_k = \pi(\mathbf{K}, \mathbf{D}, T_{I_k}^G, T_C^I, \mathbf{X_G}) \tag{4.2}$$

Here, $\pi()$ is the known camera projection model, $\mathbf{K}$ & $\mathbf{D}$ are the known camera intrinsics and distortion parameters respectively, and $\mathbf{X_G}$ is the known 3D positions of the corners on the calibration fiducial marker/target. Equation 4.2 is measurement model used in the Kalman Update step of EKF based method described in [8].

4.6 Problem Formulation

The goal is to determine the spatial 6 DoF separation, *viz.* the extrinsic calibration $T_C^I \in SE(3)$ between a camera and an IMU. As discussed in Section 3.5.1 in Chapter 3, The calibration process is divided into three steps *viz* data collection, inter sensor rotation initialization, and full extrinsic calibration.



(a) Initialization of Rotation R^I_C (b) Extended Kalman Filter for estimating both R^I_C and ${}^Ip_C$

Figure 4.3: Figure 4.3a presents the process of determination of an initial estimate of R^I_C, Figure 4.3b presents the Extended Kalman Filter which utilizes the initial estimate of R^I_C obtained in Figure 4.3a to generate both R^I_C and ${}^Ip_C$ together.

4.6.1 Data Collection

Data collection is an important step in an extrinsic calibration algorithm because the quality of calibration data determines the quality of calibration result. Similar to Section 3.5.1 in Chapter 3, the sensor suite in this Chapter involves an IMU, a proprioceptive sensor which can sense only

motion. Hence it is necessary to sufficiently excite all degrees of rotation and translation so that all the components of the extrinsic calibration parameter are completely observable.

4.6.2 Rotation Estimation

The rotation between the IMU and camera ($R_C^I \in SO(3)$) is estimated by using the rotation component of the motion based calibration constraint (Equation 4.1), given in Equation 4.3.

$$R_{I_k}^{I_{k-1}} R_C^I = R_C^I R_{C_k}^{C_{k-1}} \tag{4.3}$$

For rotation estimation, an axis angle representation of the sensor rotations has been adopted. Using the axis angle representation, Equation 4.3 can be reformulated as

$${}^{I_{k-1}}r_{I_k} = R_C^I \, {}^{C_{k-1}}r_{C_k} \tag{4.4}$$

Here ${}^{I_{k-1}}r_{I_k}$ & ${}^{C_{k-1}}r_{C_k} \in R^3$ are axis angle representations for $R_{I_k}^{I_{k-1}}$ & $R_{C_k}^{C_{k-1}}$ respectively. As shown in Figure 4.3a, checkerboard tracking is used to estimate the camera rotation $R_{C_k}^{C_{k-1}}$ between consecutive camera images. Gyroscope measurements between two image instants are integrated to estimate IMU rotation $R_{I_k}^{I_{k-1}}$. An objective function (Equation 4.5), unknown in R_C^I, is formed by squaring and summing the constraint given in Equation 4.4 for each ${}^{C_{k-1}}r_{C_k}$ and the corresponding ${}^{I_{k-1}}r_{I_k}$.

$$P = \sum_{i=1}^{M-1} \left\| {}^{I_{k-1}}r_{I_k} - R_C^I \, {}^{C_{k-1}}r_{C_k} \right\|^2 \tag{4.5}$$

In order to obtain an estimate $\hat{R}_C^I$, Equation 4.5 needs to be minimized with respect to R_C^I as shown in Equation 4.6. The Ceres [5] non linear least square solver is used to solve the optimization problem in Equation 4.6.

$$\hat{R}_C^I = \underset{R_C^I}{\operatorname{argmin}} \, P \tag{4.6}$$

The IMU rotations $R_{I_k}^{I_{k-1}}$ used to estimate R_C^I have certain shortcomings because $R_{I_k}^{I_{k-1}}$ is calcu-

lated by integrating the gyro measurements without taking the gyroscope bias into account. This step only provides an initial estimate of R_C^I which is used for initializing the EKF based full state estimation algorithm.

4.6.3 Full State Estimation using an Extended Kalman Filter

The estimation of inter sensor translation ${}^I p_C$ depends on IMU translation ${}^{I_{k-1}} p_{I_k}$ which involves double integration of IMU accelerometer measurements, but performing double integration without the knowledge of biases will introduce significant errors. An Extended Kalman Filter (EKF) is used, which, in addition to estimating R_C^I & ${}^I p_C$, also estimates the accelerometer & gyroscope biases, the pose & velocity of the IMU at the image capture instants. The block diagram for the EKF approach is shown in Figure 4.3b. The states estimated in the calibration process are:

$$\mathcal{X} = \{X^G_{I_{k=0:M-1}}, T_C^I\} \tag{4.7}$$

Here M is the number of images. $X^G_{I_k}$ is the IMU state at image timestamp k. The EKF state vector has an evolving component, and a static component. The evolving component is the IMU state at image timestamp k:

$$\hat{X}^G_{I_k} = \{{}^{I_k}_G\hat{\bar{q}}, {}^G\hat{\mathbf{v}}_{I_k}, {}^G\hat{\mathbf{p}}_{I_k}, \hat{\mathbf{b}}_{g,k}, \hat{\mathbf{b}}_{a,k}\} \tag{4.8}$$

${}^{I_k}_G\hat{\bar{q}}$ is the unit quaternion which encodes the IMU orientation such that the rotation matrix $R^\top({}^{I_k}_G\hat{\bar{q}})$ is the IMU orientation with respect to the global frame G. ${}^G\hat{\mathbf{p}}_{I_k}$ & ${}^G\hat{\mathbf{v}}_{I_k}$ are IMU position and velocity vectors ($\in R^{3\times1}$) respectively in frame G. $\hat{\mathbf{b}}_{a,k}$ & $\hat{\mathbf{b}}_{g,k}$ are the accelerometer and gyro bias vectors ($\in R^{3\times1}$) respectively. The static component of the EKF state vector is the extrinsic calibration, parameterized as T_C^I. T_C^I is formed by rotation matrix R_C^I and translation vector ${}^I\mathbf{p}_C$, however in the EKF formulation the rotation R_C^I is parameterized as a unit quaternion ${}^I_C\bar{q}$.

4.6.3.1 State Propagation

The state propagation model follows the exact implementation presented in Section 3.5.3.1 in Chapter 3, with the sole distinction that Equations 3.13 & 3.14 be replaced by Equations 4.9 & 4.10 respectively:

$$ {}^{I}_{C}\hat{\bar{q}}_{i+1} = {}^{I}_{C}\hat{\bar{q}}_{i} \tag{4.9} $$

$$ {}^{I}\hat{\mathbf{p}}_{C,i+1} = {}^{I}\hat{\mathbf{p}}_{C,i} \tag{4.10} $$

4.6.3.2 State Update

Two different state update approaches have been studied and implemented in this chapter. The first (Section 4.6.3.3) is akin to the motion based calibration technique presented in Section 3.5.3.4 in Chapter 3, although in this case the technique has been adapted for the camera sensor. The second (Section 4.6.3.4) is the author's own implementation of the method presented in [8]. One of the differences between the implementation in [8] and the author's implementation is that [8] initializes the translation variables by tape measurement and the rotation by guess an estimate close to the truth. Whereas in the author's implementation no such initial guess is required if the full estimation pipeline (Figure 4.3) is taken into consideration.

4.6.3.3 State Update using Motion Based Extrinsic Calibration

This State Update module requires the knowledge of a measurement model, measurement residual and the measurement Jacobians with respect to the state variables. This section presents the measurement model and the measurement residual, but the derivation of measurement Jacobians has been omitted in the interest of space (Detials about calculating Jacobians given in Chapter A, Sections A.1 & A.2). The camera pose from checkerboard tracking, parameterized as $T_{C_k}^{C_{k-1}} \in SE(3)$, is used as measurement. The *motion based calibration* constraint in Equation 4.1 has been used to derive the measurement model. Manipulating Equation 4.1 gives the measurement

model as presented in Equation 4.11:

$$T_{C_k}^{C_{k-1}} = (T_C^I)^{-1}(T_{I_{k-1}}^G)^{-1}T_{I_k}^G T_C^I \tag{4.11}$$

The LHS of Equation 4.11 is the measurement and the RHS is a function of state variables T_C^I, $T_{I_{k-1}}^G$, $T_{I_k}^G$. So, the measurement model (Equation 4.11) is in agreement with the standard form $z = h(x)$ used in EKF, where z is the measurement and $h()$ is the measurement model which is a function of the state x. In the current scenario, measurement $z = T_{C_k}^{C_{k-1}}$ & measurement model $h(x) = (T_C^I)^{-1}(T_{I_{k-1}}^G)^{-1}T_{I_k}^G T_C^I$ and state $x = \{T_{I_{k-1}}^G, T_{I_k}^G, T_C^I\}$. Here,

$$T_C^I = \begin{bmatrix} \mathbf{R}({}_C^I\bar{q}) & {}^I\mathbf{p}_C \\ 0 & 1 \end{bmatrix}, T_{I_{k-1}}^G = \begin{bmatrix} \mathbf{R}^T({}_G^{I_{k-1}}\bar{q}) & {}^G\mathbf{p}_{I_{k-1}} \\ 0 & 1 \end{bmatrix}$$

$$T_{I_k}^G = \begin{bmatrix} \mathbf{R}^T({}_G^{I_k}\bar{q}) & {}^G\mathbf{p}_{I_k} \\ 0 & 1 \end{bmatrix}$$

Clearly T_C^I, $T_{I_{k-1}}^G$, $T_{I_k}^G$ depend on state variables.

Separating the rotation and translation components of Equation 4.11, one obtains.

$$\begin{aligned} R_{C_k}^{C_{k-1}} &= \mathbf{R}^\top({}_C^I\bar{q})\mathbf{R}({}_G^{I_{k-1}}\bar{q})\mathbf{R}^T({}_G^{I_k}\bar{q})\mathbf{R}({}_C^I\bar{q}) \\ {}^{C_{k-1}}p_{C_k} &= \mathbf{R}^\top({}_C^I\bar{q})\Bigg[\left(\mathbf{R}({}_G^{I_{k-1}}\bar{q})\mathbf{R}^T({}_G^{I_k}\bar{q}) - \mathbf{I}\right){}^I\mathbf{p}_C \\ &\quad + \mathbf{R}({}_G^{I_{k-1}}\bar{q})\left({}^G\mathbf{p}_{I_k} - {}^G\mathbf{p}_{I_{k-1}}\right)\Bigg] \end{aligned} \tag{4.12}$$

The measurement model calculated at state estimates gives the predicted rotation and transla-

tion measurement, *viz.* $\hat{R}_{C_k}^{C_{k-1}}$ and ${}^{C_{k-1}}\hat{p}_{C_k}$ respectively. The difference between the true measurements and predicted measurements gives the measurement residual $\mathbf{r}_k$ required for state update (Equation 4.13).

$$\mathbf{r}_k = \begin{bmatrix} \texttt{Log}(R_{C_k}^{C_{k-1}}(\hat{R}_{C_k}^{C_{k-1}})^\top) \\ {}^{C_{k-1}}p_{C_k} - {}^{C_{k-1}}\hat{p}_{C_k} \end{bmatrix} \tag{4.13}$$

Here, `Log()` associates a matrix $R \in SO(3)$ to a vector $\in R^{3\times1}$ (via a skew symmetric matrix). In addition to the measurement residual $\mathbf{r}_k$, one also requires the Jacobians of the measurement model with respect to the state variables in order to perform state and covariance update. The Jacobians (Equation 4.14) are evaluated at the best available estimate of the state variables $x = \{T_{I_{k-1}}^G, T_{I_k}^G, T_C^I\}$.

$$\begin{aligned} \mathbf{H}_{T_C^I}^{T_{C_k}^{C_{k-1}}} &= \frac{\partial T_{C_k}^{C_{k-1}}}{\partial T_C^I}\bigg|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}} \\ \mathbf{H}_{T_{I_{k-1}}^G}^{T_{C_k}^{C_{k-1}}} &= \frac{\partial T_{C_k}^{C_{k-1}}}{\partial T_{I_{k-1}}^G}\bigg|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}} \\ \mathbf{H}_{T_{I_k}^G}^{T_{C_k}^{C_{k-1}}} &= \frac{\partial T_{C_k}^{C_{k-1}}}{\partial T_{I_k}^G}\bigg|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}} \end{aligned} \tag{4.14}$$

4.6.3.4 State Update using Re-projection Based Extrinsic Calibration

This State Update module requires the detection of checkerboard corners on the calibration checkerboard target and uses it as measurement, instead of using the sensor pose as measurement as done in Sections 3.5.3.4 of Chapter 3, 4.6.3.3 of the current chapter and in [62]. The practice of using checkerboard corner pixels as measurement has been adopted in [7] and [8]. This section is inspired from the technique presented in [8]. It is to be noted here that the 3D position $\mathbf{X_G}$ of these detected corners is known. Equation 4.5 is used as the measurement model where z_k are the corner pixel measurements at time k of the known 3D positions $\mathbf{X_G}$. The measurement model $\pi()$

(RHS of Equation 4.2) is a function of the unknown state variables like the extrinsic calibration parameter T_C^I and the IMU pose $T_{I_k}^G$.

The re-projection based measurement model (Equation 4.2) calculated at best available state estimates gives the predicted corner pixel locations denoted as $\pi(\mathbf{K}, \mathbf{D}, \hat{T}_{I_k}^G, \hat{T}_C^I, \mathbf{X_G})$. The difference between the true measurements z_k and predicted measurements gives the measurement residual $\mathbf{r}_k$ required for state update (Equation 4.15).

$$\mathbf{r}_k = z_k - \pi(\mathbf{K}, \mathbf{D}, \hat{T}_{I_k}^G, \hat{T}_C^I, \mathbf{X_G}) \tag{4.15}$$

The measurement Jacobians (Equation 4.16) are evaluated at the best available estimate of the state variables $x = \{T_{I_{k-1}}^G, T_{I_k}^G, T_C^I\}$.

$$\begin{aligned}
\mathbf{H}_{T_C^I}^{z_k} &= \left.\frac{\partial z_k}{\partial T_C^I}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}} \\
\mathbf{H}_{T_{I_{k-1}}^G}^{z_k} &= \left.\frac{\partial z_k}{\partial T_{I_{k-1}}^G}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}} \\
\mathbf{H}_{T_{I_k}^G}^{z_k} &= \left.\frac{\partial z_k}{\partial T_{I_k}^G}\right|_{\hat{x}=\{\hat{T}_{I_{k-1}}^G, \hat{T}_{I_k}^G, \hat{T}_C^I\}}
\end{aligned} \tag{4.16}$$

The details concerning calculating Jacobians can be found in Chapter A, Sections A.1 & A.3.

4.6.3.5 Update Equations

Depending on the type of measurement model used (Section 4.6.3.3 or 4.6.3.4), the individual Jacobians are stacked together to form a consolidated Jacobian $\mathbf{H}_k$. Along with the respective measurement residual $\mathbf{r}_k$, the consolidated Jacobian $\mathbf{H}_k$ are used for state and state-covariance update when a measurement update is available. The update equations are presented in Equations

4.17-4.19.

$$\mathbf{K}_k = \mathbf{P}_{k\_}\mathbf{H}_k^\top(\mathbf{H}_k\mathbf{P}_{k\_}\mathbf{H}_k^\top + \mathbf{R})^{-1} \tag{4.17}$$

$$\begin{bmatrix} \hat{X}^G_{I_{k+}} \\ \hat{T}^I_{C+} \end{bmatrix} = \begin{bmatrix} \hat{X}^G_{I_{k-}} \\ \hat{T}^I_{C-} \end{bmatrix} \oplus \mathbf{K}_k\mathbf{r}_k \tag{4.18}$$

$$\mathbf{P}_{k_+} = \mathbf{P}_{k_-} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k_-} \tag{4.19}$$

Here $\mathbf{K}_k$ is the Kalman gain which is used in Equations 4.18 and 4.19 for state and state covariance update respectively. $\mathbf{R}$ is the tunable measurement covariance matrix. '-' denotes the estimate before update while '+' denotes the estimate after update. $\oplus$ in Equation 4.18 refers to generic composition which can be algebraic addition for variables on vector space or rotation composition for variables on $SO(3)$.

4.7 System Description

Data for experiments was collected from a sensor system (Figure 4.1) comprising a Basler Ace Camera $[1600 \times 1200]$ running at 30 Hz, and a Vectornav VN-300 Camera running at 400 Hz.

4.8 Experiments

Experiments were performed on 5 different datasets collected by motion exciting the sensor suite (Figure 4.1) along all translation and about all rotational degrees of freedom. Both methods of performing EKF update *viz.* the motion based calibration approach (henceforth called **MoCal**), and the reprojection error based calibration (henceforth called **RepErrCal**) approach were evaluated and their results were compared against another popular and widely used open-sourced camera-IMU calibration algorithm called Kalibr[7][3]. At present, Kalibr[7] happens to be the only open-sourced Camera-IMU calibration toolbox available to the robotics community.

[3] `https://github.com/ethz-asl/kalibr/wiki/camera-imu-calibration`

Results of camera IMU calibration when **motion based calibration residual** is used for EKF update							
Dataset No.	Method	ϕ°	θ°	ψ°	$\mathbf{X}[\mathbf{cm}]$	$\mathbf{Y}[\mathbf{cm}]$	$\mathbf{Z}[\mathbf{cm}]$
1	Kalibr	88.9694	1.0825	90.8270	7.3282	12.5850	-7.4703
	MoCal	89.9676	1.4810	91.1667	5.9607	13.0148	-7.7642
	Difference	-0.9919	-0.3399	0.3983	1.0968	-0.3775	0.1677
2	Kalibr	88.9854	1.0673	90.8232	7.3315	12.6650	-7.5365
	MoCal	89.9248	0.9998	89.1244	6.3983	13.1455	-7.6399
	Difference	-0.9710	1.6986	-0.0657	0.9411	-0.3653	0.3047
3	Kalibr	89.0208	1.0706	90.7984	7.3344	12.5920	-7.5386
	MoCal	89.4866	1.2630	89.1469	5.5976	12.8140	-7.8559
	Difference	-0.4964	1.6495	0.2067	1.8528	-0.2033	0.4289
4	Kalibr	89.0547	1.0927	90.8185	7.2727	12.5420	-7.5599
	MoCal	90.8153	1.3007	89.6592	5.5994	12.6143	-8.0560
	Difference	-1.7828	1.1619	0.1913	1.4462	0.0761	0.5659
5	Kalibr	89.0552	1.0770	90.8315	7.3027	12.5560	-7.5138
	MoCal	89.6153	1.1471	90.2770	5.9305	12.8446	-7.6965
	Difference	-0.5705	0.5540	0.0738	1.3190	-0.2396	0.2232

Table 4.1: Tabulation of T_C^I (camera imu calibration) when the motion based calibration residual is used in the EKF update step. The results from Kalibr camera IMU calibration is used as a reference for comparison. Although the difference in rotation about all axes is less than $\approx 2^\circ$, the difference along $\mathbf{X}$-axis is > 0.9 cm

4.9 Discussion

4.9.1 Motion Based Calibration

The results obtained from motion based calibration approach (MoCal) has been presented in Table 4.1. In addition to that, the results from Kalibr and the difference between Kalibr and MoCal has also been presented for the sake of comparison. For all the datasets experimented with, although the difference in rotation about all axes is less than $\approx 2^\circ$, the difference along $\mathbf{X}$-axis is > 0.9 cm.

4.9.2 Re-projection Error Minimization Based Calibration

The results obtained from reprojection error minimization based calibration approach (RepErrCal) has been presented in Table 4.2. In addition to that, the results from Kalibr and the difference between Kalibr and RepErrCal has also been presented for the sake of comparison. For all the

Results of camera IMU calibration when **re-projection error based residual** is used for EKF update							
Dataset No.	Method	ϕ°	θ°	ψ°	$\mathbf{X}[\mathbf{cm}]$	$\mathbf{Y}[\mathbf{cm}]$	$\mathbf{Z}[\mathbf{cm}]$
1	Kalibr	88.9694	1.0825	90.8270	7.3282	12.5850	-7.4703
	RepErrCal	88.9471	1.0832	90.9106	7.2580	12.4892	-7.7670
	Difference	0.0238	-0.0836	-0.0008	0.0640	0.0929	0.2863
2	Kalibr	88.9854	1.0673	90.8232	7.3315	12.6650	-7.5365
	RepErrCal	89.0770	1.0407	90.7982	7.2112	12.7040	-8.0458
	Difference	-0.0921	0.0255	-0.0262	0.1034	-0.0237	0.5183
3	Kalibr	89.0208	1.0706	90.7984	7.3344	12.5920	-7.5386
	RepErrCal	88.9734	1.0444	90.8052	7.1492	12.5221	-8.1087
	Difference	0.0476	-0.0063	-0.0263	0.1947	0.0677	0.5750
4	Kalibr	89.0547	1.0927	90.8185	7.2727	12.5420	-7.5599
	RepErrCal	89.1348	1.0712	90.7966	6.9787	12.3535	-7.9911
	Difference	-0.0804	0.0222	-0.0212	0.2798	0.2013	0.4385
5	Kalibr	89.0552	1.0770	90.8315	7.3027	12.5560	-7.5138
	RepErrCal	89.0694	1.0830	90.9919	6.7944	12.5258	-7.7120
	Difference	-0.0112	-0.1604	0.0034	0.4843	0.0310	0.1784

Table 4.2: Tabulation of T_C^I (camera imu calibration) when the re-projection error based calibration residual is used in the EKF update step. The results from Kalibr camera IMU calibration is used as a reference for comparison. The maximum difference in rotation about all axes is 0.1604°, and the maximum difference along all translation direction is 0.5750 cm.

datasets experimented with, the maximum difference in rotation about all axes is 0.1604°, and the maximum difference along all translation direction is 0.5750 cm.

4.9.3 Comparison with respect to Kalibr

The difference between MoCal/RepErrCal and Kalibr[7] has been presented in Table 4.3 from which it can be concluded that the difference between MoCal and Kalibr is in general larger than the difference between RepErrCal and Kalibr. To be specific, the rotation difference between MoCal and Kalibr is always larger than the rotation difference between RepErrCal and Kalibr, and although the difference along **Z**-axis between RepErrCal and Kalibr is slightly higher (by 2.136 mm at maximum, for Dataset 2) for most datasets than the corresponding difference between MoCal and Kalibr, the distance difference ($\sqrt{\mathbf{X}^2 + \mathbf{Y}^2 + \mathbf{Z}^2}$ $[\mathbf{cm}]$) between MoCal and Kalibr is always larger (by 5.255 mm at minimum, for Dataset 2) than the distance difference between RepErrCal and Kalibr.

Difference in estimation of T_C^I with respect to Kalibr								
Dataset No.	Method	ϕ°	θ°	ψ°	$\mathbf{X}[\mathbf{cm}]$	$\mathbf{Y}[\mathbf{cm}]$	$\mathbf{Z}[\mathbf{cm}]$	$\sqrt{\mathbf{X}^2+\mathbf{Y}^2+\mathbf{Z}^2}\ [\mathbf{cm}]$
1	MoCal	-0.9919	-0.3399	0.3983	1.0968	-0.3775	0.1677	1.1720
	RepErrCal	0.0238	-0.0836	-0.0008	0.0640	0.0929	0.2863	0.3077
2	MoCal	-0.9710	1.6986	-0.0657	0.9411	-0.3653	0.3047	1.0545
	RepErrCal	-0.0921	0.0255	-0.0262	0.1034	-0.0237	0.5183	0.5290
3	MoCal	-0.4964	1.6495	0.2067	1.8528	-0.2033	0.4289	1.9126
	RepErrCal	0.0476	-0.0063	-0.0263	0.1947	0.0677	0.5750	0.6104
4	MoCal	-1.7828	1.1619	0.1913	1.4462	0.0761	0.5659	1.5548
	RepErrCal	-0.0804	0.0222	-0.0212	0.2798	0.2013	0.4385	0.5578
5	MoCal	-0.5705	0.5540	0.0738	1.3190	-0.2396	0.2232	1.3590
	RepErrCal	-0.0112	-0.1604	0.0034	0.4843	0.0310	0.1784	0.5170

Table 4.3: Difference between results from motion based calibration update/re-projection error based calibration update and Kalibr.

Average Re-projection Error		
Dataset No.	MoCal	RepErrCal
1	18.91	2.70
2	35.18	2.76
3	38.22	2.99
4	33.23	2.86
5	15.06	2.29

Table 4.4: Average Re-projection error with motion-based calibration residual and re-projection residual.

4.9.4 Comparison of Average Reprojection Errors

The average re-projection errors of the known 3D checkerboard corner locations w.r.t the corresponding detected checkerboard corner pixels is used as a metric to evaluate calibration accuracy between MoCal and RepErrCal. The re-projection error, given in Equation 4.15, is a function of the camera IMU extrinsic calibration parameter T_C^I, which means that a higher reprojection error may imply that the extrinsic calibration parameter is farther from the true value. The average re-projection errors for both the competing methods MoCal, RepErrCal have been presented in Table 4.4 where it can be seen that with MoCal the average reprojection error is higher. The reprojection errors have been pictorially presented in Figure 4.4. For MoCal they have been presented in Figures 4.4a, 4.4c, 4.4e, 4.4g, and for RepErrCal in Figures 4.4b, 4.4d, 4.4f, 4.4h. In Figures 4.4a,

4.4c, 4.4e, 4.4g, one can see that the corners of the checkerboard target do not coincide with the re-projected 3D checkerboard corner (magenta) pixel positions, whereas in Figures 4.4b, 4.4d, 4.4f, 4.4h the mis-alignment between the actual checkerboard corners and the 3D projected corners' pixel locations is not visible as evident from a lower re-projection error.

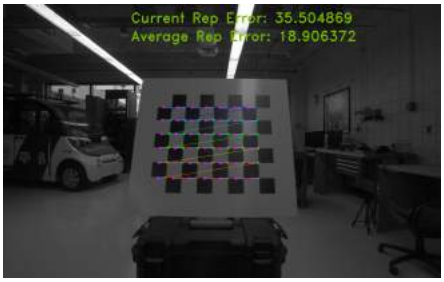
4.10 Conclusion

As introduced in Section 1.4 of Chapter 1, this chapter presented two different implementations of Camera IMU registration using an Extended Kalman Filter based estimation framework when the EKF update step was formulated in two different ways, *viz.* one which considers the exteroceptive sensor (camera in this case) as a pose sensor (similar to [62] & Chapter 3) and thus exploits a motion based calibration constraint [18] for Camera-IMU extrinsic calibration, and second which uses the minimization of re-projection error between detected checkerboard corners and the projection of the corresponding known 3D points onto the image plane as done in [8]. The first method is called MoCal while the second method is named RepErrCal. This chapter presented a comparison between these two methods. This comparison not only helped to know which approach estimates better results but also helped to choose the method to be used in Joint Calibration of LiDAR-IMU-Camera sensor in Chapter 5. From the observations from experiments in Section 4.8, it can be concluded that RepErrCal estimates results closer to the widely used camera-IMU toolbox named Kalibr as evident from the low difference in estimates presented in Tables 4.2 & 4.3, and also results in lower reprojection error better than MoCal (Table 4.4 and Figure 4.4) during verification which can be attributed to the fact that the MoCal uses camera as pose sensor which may be subjected to drift during pose estimation where as RepErrCal minimizes the reprojection error of known 3D points.

As far as a broad picture of sensor registration is concerned, the EKF based method estimates static IMU-Camera registration and the evolving IMU pose and velocity, quite akin to a multi-sensor SLAM/Odometry algorithm, albiet with the aid of a calibration target. The method serves as a baseline or a foundation for developing generic online multisensor calibration, localization and mapping algorithms where individual sensor measurements can be used to register them with

respect to each other and also perform their registration with respect to a fixed frame of reference.

In the next Chapter 5, all the topics discussed in Chapters 2-4 will be utilized to perform joint extrinsic calibration of a LiDAR-IMU-camera sensor system with the objective to not only show that joint calibration of multiple sensors yields superior results when compared against individual pairwise calibration of sensors but also to lay the foundation for a generic pose estimation framework which uses measurements from multiple sensors to estimate static registration parameters between rigidly mounted sensors and evolving registration parameters with respect to a fixed frame of refrence. Moreover, Chapter 5 is the first contribution in the sensor registration literature that proposes joint optimization of a LiDAR-IMU-camera system to estimate various variables of interest, the static registration parameter being the most important variable of concern in the current context.



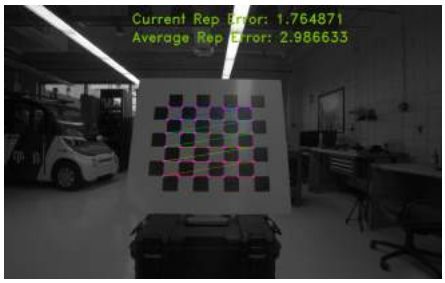
(a) Average Re-projection Error (=18.91) with Motion Based Calibration Residual for **Dataset 1**.

(b) Average Re-projection Error with Re-projection (=2.70) Based Calibration Residual for **Dataset 1**.

(c) Average Re-projection Error (=35.18) with Motion Based Calibration Residual for **Dataset 2**.

(d) Average Re-projection Error (=2.76) with Re-projection Based Calibration Residual for **Dataset 2**.

(e) Average Re-projection Error (=38.22) with Motion Based Calibration Residual for **Dataset 3**.



(f) Average Re-projection Error (=2.99) with Re-projection Based Calibration Residual for **Dataset 3**.

(g) Average Re-projection Error (=33.23) with Motion Based Calibration Residual for **Dataset 4**.

(h) Average Re-projection Error (=2.86) with Re-projection Based Calibration Residual for **Dataset 4**.

Figure 4.4: Comparison of Average Re-projection Error with the motion based (MoCal) and re-projection error based calibration (RepErrCal) residuals. With MoCal, the pixel projection (magenta) of known 3D checkerboard corner positions is far from the actual checkerboard corner locations, unlike RepErrCal.

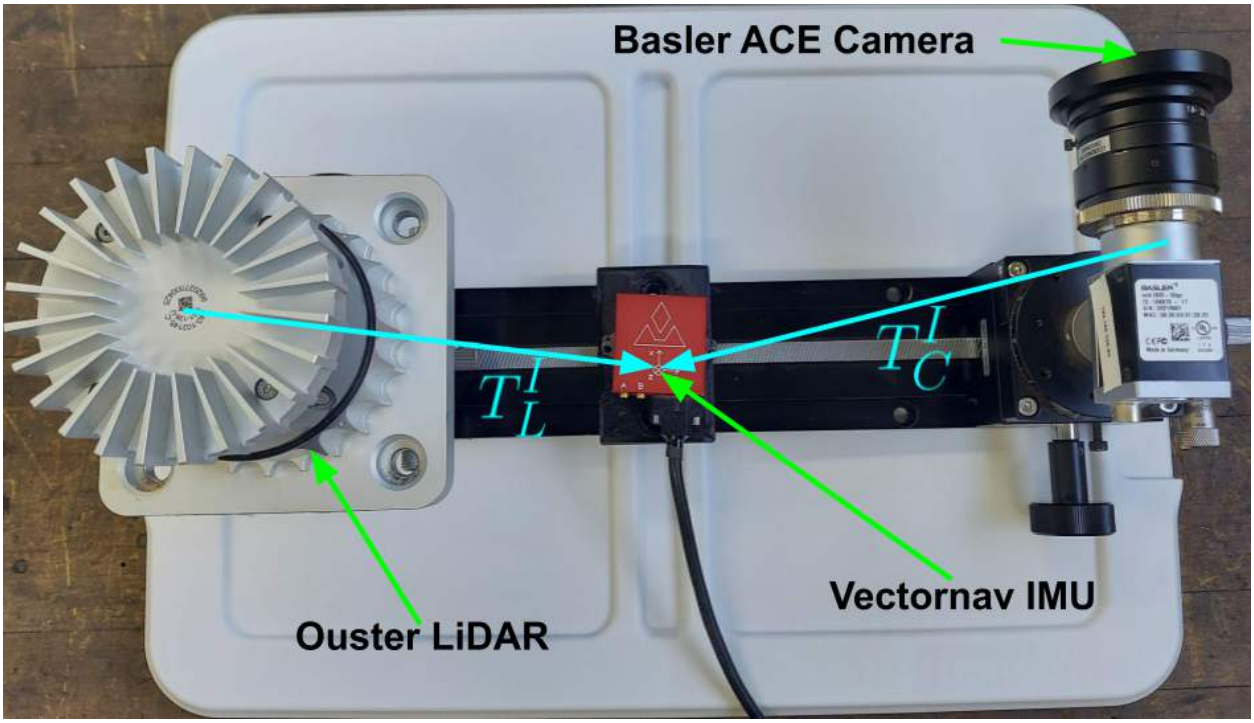


Figure 5.1: **Lidar IMU Camera System:** Consists an Ouster 128 Channel LiDAR running at 10 Hz, a Basler Ace Camera $[1600 \times 1200]$ running at 30 Hz, and a Vectornav VN-300 Camera running at 400 Hz. The goal of the joint extrinsic calibration algorithm is to estimate LiDAR-IMU extrinsic calibration T_L^I and the camera-IMU extrinsic calibration T_C^I.

5.1 Introduction

Pair-wise registration of Camera-IMU systems [8], [7], [62], [63] is a well studied problem owing to the popularity of Visual Inertial Odometry and Visual-Inertial SLAM algorithms developed over the last decade. Similarly, pair-wise registration of LiDAR-IMU systems [1], [16], Chapter 3 is also a problem which has seen some contributions in literature (although not as much as Camera-IMU registration) in the last 5 years, after the popularity of LOAM [2] and other Lidar Inertial State estimation algorithms [64], [54]. Recently, there have been two contributions on Lidar-Visual-

Inertial state estimation [53], [65] which assume that the LiDAR-Inertial and the Visual-Inertial extrinsic calibration parameters are known *a-priori* and authors perform individual pair-wise calibration of LiDAR-IMU and Camera-IMU systems in order to run the Lidar-Visual-Inertial state estimation framework. In the author's opinion, since the sensors are being used together for state estimation, they should be registered or calibrated together in a joint fashion using all pair-wise measurement constraints (LiDAR↔IMU, IMU↔Camera, LiDAR↔Camera) together in order to get the best registration parameters between sensor pairs.

5.2 Literature Review

There isn't any contribution in literature which does joint calibration of a LiDAR, IMU, Camera system by using measurement residuals between all the three pairs (LiDAR-IMU, Camera-IMU, LiDAR-Camera). Although [20] attempts to calibrate LiDAR, IMU and camera together, it does so by performing camera IMU calibration first and then cross calibrating the camera IMU system with the LiDAR, so by definition it does not perform joint calibration of all the sensors together. Moreover [20] involves a 2D-LiDAR, a stereo camera and an IMU, so it deals with a system much different from the one addressed in this chapter where a 3D-LiDAR, a monocular camera and an IMU are registered (Figure 5.1).

5.3 Contribution

Although there are several contributions on LiDAR-Camera, Camera-IMU, LiDAR-IMU pair-wise calibration, contributions on LiDAR-IMU-Camera joint calibration is virtually non-existent in literature. This work presents a LiDAR-IMU-Camera joint calibration approach which derives motivation from previous approaches on LiDAR-Camera ([24], *surface point to plane constraint* described in Chapter 2 Section 2.4.3.1), LiDAR-IMU (Chapter 3), Camera-IMU ([8], Chapter 4) extrinsic calibration, and performs joint extrinsic calibration of these three sensors under an EKF framework. The proposed method requires a calibration checkerboard in the common field of view of the LiDAR and camera inorder to ease the challenge of feature association across a camera LiDAR pair. Although the primary focus is to jointly estimate the static registration parameters

between LiDAR & IMU sensor and Camera & IMU sensor, the framework also estimates evolving IMU poses and velocities as done in Chapter 3 & 4 which also use a EKF framework like the method proposed in this chapter does. So, if the challenge of feature data (planes & lines, etc.) association between Camera and LiDAR is addressed, the proposed framework can be used for generic sensor registration i.e. simultaneous multi-sensor pose estimation and LiDAR+visual mapping.

5.4 Problem Formulation

The goal is to determine the spatial 6 DoF separation, *viz.* the extrinsic calibration $T_L^I \& T_C^I \in SE(3)$ between an IMU and a LiDAR/Camera (Figure 5.1), by simultaneously exploiting measurement constraints between all the sensor pairs under an EKF based estimation framework. Similar to the LiDAR-IMU registration and Camera-IMU registration methods described in the previous Chapters (3, 4), the calibration process is divided into three steps *viz.* data collection, inter sensor rotation initialization, and full extrinsic calibration using an Extended Kalman Filter (EKF).

5.4.1 Data Collection

Data collection is an important step in an extrinsic calibration algorithm because the quality of calibration data determines the quality of calibration result. Similar to Section 3.5.1 & Section 4.6.1 in Chapter 3 & Chapter 4 respectively, the sensor suite in this Chapter also involves an IMU, a proprioceptive sensor which can sense only motion. Hence it is necessary to sufficiently excite all degrees of rotation and translation so that all the components of the extrinsic calibration parameter are completely observable. However, in addition to this, one must also keep in mind that the camera and the LiDAR sensors need to share a common field of view so that they will be able to detect a calibration target (Figure 5.2) which will be used to form an additional measurement constraint that jointly updates both the LiDAR-IMU & Camera-IMU calibration estimates at the same time.

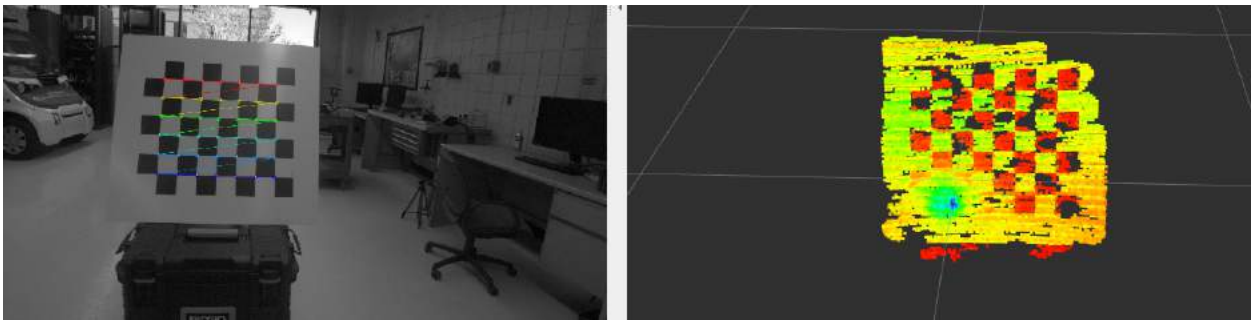
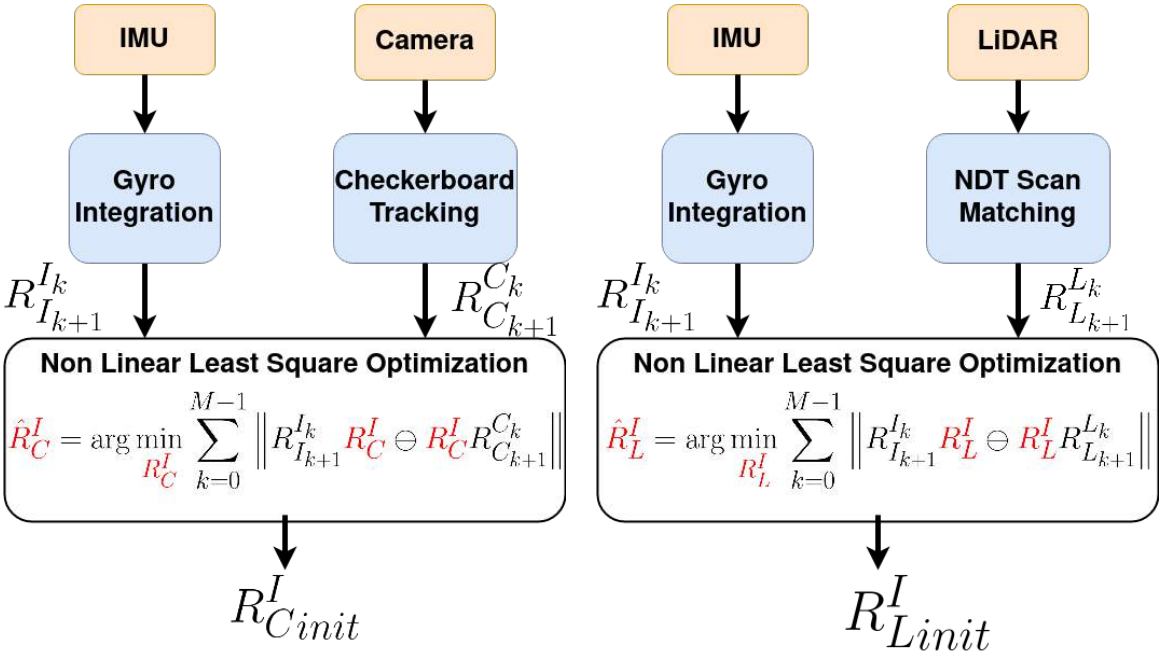


Figure 5.2: Simultaneous detection of calibration checkerboard target in Camera and LiDAR.

5.4.2 Inter Sensor Rotation Initialization



(a) Initialization of Rotation $R^I_C \in SO(3)$ between Camera and IMU

(b) Initialization of Rotation $R^I_L \in SO(3)$ between LiDAR and IMU

Figure 5.3: Initialization of rotation components of T^I_C & T^I_L

The details of rotation initialization for Camera-IMU & LiDAR-IMU are presented in Sections 3.5.2 & 4.6.2 respectively which has been omitted here for brevity. However, the overview of the

rotation initialization process is presented in Figure 5.3.

5.4.3 Full State Estimation using an Extended Kalman Filter

In addition to the measurement residuals previously used in Chapter 3 and Chapter 4 to estimate T_L^I and T_C^I pairs respectively, the detection of a rectangular checkerboard calibration target in both LiDAR and camera to further constrain the optimization by introducing LiDAR-camera measurement residuals ([24], *surface point to plane constraint* described in Chapter 2 Section 2.4.3.1) is also utilized under an EKF framework to perform joint calibration of the LiDAR IMU Camera system. In addition to the static extrinsic calibration parameters T_L^I & T_C^I, the EKF also estimates the time-varying accelerometer & gyroscope biases ($\hat{\mathbf{b}}_{a,k}$, $\hat{\mathbf{b}}_{g,k} \in R^{3\times1}$), the pose ($T_{I_k}^G \in SE(3)$) & velocity (${}^G\hat{\mathbf{v}}_{I_k} \in R^{3\times1}$) of the IMU. The states estimated are:

$$\mathcal{X} = \{X_{I_{k=0:M-1}}^G, T_L^I, T_C^I\} \tag{5.1}$$

In Chapter 3, T_L^I formed a part of the estimated state variables and T_C^I was estimated in Chapter 4, in the current Chapter both T_L^I&T_C^I are estimated together. Here M is the total number of measurements from the LiDAR and the Camera. $X_{I_k}^G$ is the IMU state at LiDAR scan/image timestamp k. The EKF state vector has an evolving component and a static component. The evolving component is the IMU state at LiDAR scan/image timestamp k:

$$\hat{X}_{I_k}^G = \{{}_G^{I_k}\hat{\bar{q}}, {}^G\hat{\mathbf{v}}_{I_k}, {}^G\hat{\mathbf{p}}_{I_k}, \hat{\mathbf{b}}_{g,k}, \hat{\mathbf{b}}_{a,k}\} \tag{5.2}$$

${}_G^{I_k}\hat{\bar{q}}$ is the unit quaternion which encodes the IMU orientation such that the rotation matrix $R^\top({}_G^{I_k}\hat{\bar{q}})$ is the IMU orientation with respect to the global frame G. The static components of the EKF state vector are the extrinsic calibration parameters, parameterized as $T_C^I \in SE(3)$ & $T_L^I \in SE(3)$. Under the EKF formulation the rotation $R \in SO(3)$ is parameterized as a unit quaternion $\bar{q}$. The block diagram of the proposed approach is presented in Figure 5.4.

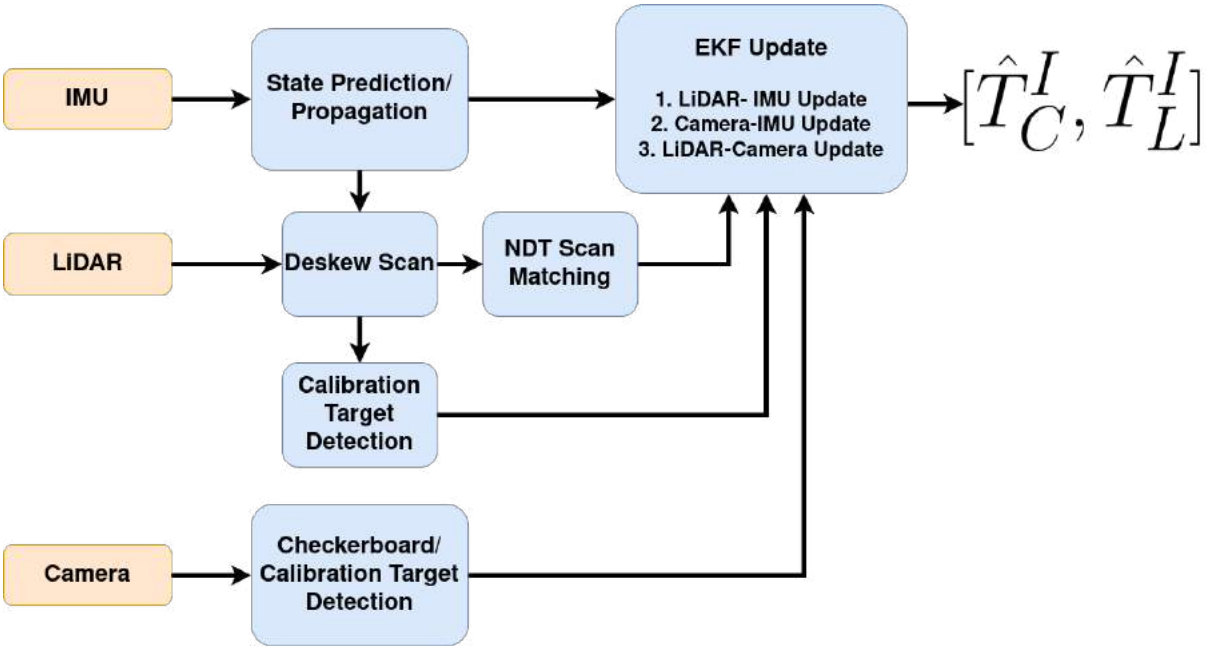


Figure 5.4: This Figure presents the Extended Kalman Filter which utilizes the initial estimates of R_L^I & R_C^I obtained in Figure 3.5a & 4.3a respectively to estimate both $T_L^I = \begin{bmatrix} R_L^I & {}^I p_L \\ 0 & 1 \end{bmatrix}$ & $T_C^I = \begin{bmatrix} R_C^I & {}^I p_C \\ 0 & 1 \end{bmatrix}$ together.

5.4.3.1 State Propagation

The state propagation module is similar to the ones presented in Sections 3.5.3.1 and 4.6.3.1 used for estimation of T_L^I and T_C^I respectively in an individual manner. In the current context T_L^I and T_C^I are estimated together which requires that both T_L^I and T_C^I are part of the state vector which justifies Equations 5.8 - 5.11. As in Sections 3.5.3.1 and 4.6.3.1, the discrete time implementation given in [8], [57] are used to propagate the EKF state (Equations 5.3 - 5.11) from IMU timestamp i to $i+1$. The gyroscope and accelerometer measurements $\boldsymbol{\omega}_{m,i} \in R^{3\times1}$ & $\mathbf{a}_{m,i} \in R^{3\times1}$ respectively, are assumed to be constant during the IMU sampling period Δt. In the following equations ${}^G\mathbf{g}$ is the acceleration due to gravity in the global frame G.

$$
{}^{I_{i+1}}_{G}\hat{\bar{q}} = \exp\left(\frac{1}{2}\Omega(\boldsymbol{\omega}_{m,i} - \hat{\mathbf{b}}_{g,i})\Delta t\right) {}^{I_i}_{G}\hat{\bar{q}} \tag{5.3}
$$

$$
{}^{G}\hat{\mathbf{v}}_{I_{i+1}} = {}^{G}\hat{\mathbf{v}}_{I_i} - {}^{G}\mathbf{g}\Delta t + \hat{\mathbf{R}}^{G}_{I_i}(\mathbf{a}_{m,i} - \hat{\mathbf{b}}_{a,i})\Delta t \tag{5.4}
$$

$$
\begin{aligned}
{}^{G}\hat{\mathbf{p}}_{I_{i+1}} = {}^{G}\hat{\mathbf{p}}_{I_i} + {}^{G}\hat{\mathbf{v}}_{I_i}\Delta t - \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2 \\
+ \frac{1}{2}\hat{\mathbf{R}}^{G}_{I_i}(\mathbf{a}_{m,i} - \hat{\mathbf{b}}_{a,i})\Delta t^2
\end{aligned} \tag{5.5}
$$

$$
\hat{\mathbf{b}}_{g,i+1} = \hat{\mathbf{b}}_{g,i} \tag{5.6}
$$

$$
\hat{\mathbf{b}}_{a,i+1} = \hat{\mathbf{b}}_{a,i} \tag{5.7}
$$

$$
{}^{I}_{L}\hat{\bar{q}}_{i+1} = {}^{I}_{L}\hat{\bar{q}}_i \tag{5.8}
$$

$$
{}^{I}\hat{\mathbf{p}}_{L,i+1} = {}^{I}\hat{\mathbf{p}}_{L,i} \tag{5.9}
$$

$$
{}^{I}_{C}\hat{\bar{q}}_{i+1} = {}^{I}_{C}\hat{\bar{q}}_i \tag{5.10}
$$

$$
{}^{I}\hat{\mathbf{p}}_{C,i+1} = {}^{I}\hat{\mathbf{p}}_{C,i} \tag{5.11}
$$

More details about the state and covariance propagation can be obtained from Sections 3.5.3.1 & 4.6.3.1 which has been skipped here for brevity.

5.4.3.2 State Update

The state/covariance update process is peformed when 1. The camera captures an image with the calibration/checkerboard target, or 2. The LiDAR captures a LiDAR scan, or 3. The calibration target is detected both in the camera image and the LiDAR scan. While 1 & 2 perform individual estimation of T_C^I & T_L^I respectively, 3 performs joint estimation T_C^I & T_L^I.

5.4.3.3 Camera - IMU Update

The details of Camera-IMU state update can be found in Chapter 4 (and in [8]). As described in Chapter 4, camera-IMU extrinsic calibration can be done using a motion based calibration technique (Section 4.4) or a reprojection error minimization based approach (Section 4.5). In the current context the reprojection error based approach is preferred owing to its superior performance

as described in Section 4.8 from comparing average reprojection errors between the motion based aproach and the reprojection error based approach (in Table 4.4 & Figure 4.4). In the re-projection error minimization based approach the detected corners z_{IC} of the calibration checkerboard (detected using OpenCV[49] in the incoming image as shown in block diagram of Figure 5.4) are used as measurements to update IMU-Camera extrinsic calibration T_C^I and other IMU states. The method involves calculation of measurement residuals $r_{IC} = z_{IC} - h(T_{I_k}^G, T_C^I, \mathbf{X}_G)$ and the corresponding Jacobians $H^{z_{IC}}_{T_{I_k}^G, T_C^I}$ for state & covariance update. $h()$ is the camera projection model used to project the known 3D positions $\mathbf{X}_G$ of the checkerboard's corners using the best available estimates of $T_{I_k}^G, T_C^I$.

5.4.3.4 LiDAR - IMU Update

The details of LiDAR-IMU state update can be found in Chapter 3 Section 3.5.3.4. Briefly, as shown in Figure 5.4, IMU measurements and the best available estimate of extrinsic calibration T_L^I are used to remove motion distortion (Section 3.5.3.2) from the LiDAR scan before performing scan matching for LiDAR pose estimation (Section 3.5.3.3). Ego motion distortion, which happens because of the sequential nature of measurement acquisition in spinning LiDARs, occurs during calibration data collection when the user motion excites the sensor suite along all directions, about all axes. The undistorted scans are used for scan matching based LiDAR pose estimation with the goal to perform motion based calibration [18] under an EKF paradigm. The measurement z_{IL} in this case is the LiDAR pose resulting from scan matching, and the predicted measurement is $h(X_{I_k}^G, T_L^I) \in SE(3)$. The resulting measurement residuals $r_{IL} = z_{IL} \ominus h(X_{I_k}^G, T_L^I)$ and the corresponding Jacobians $H^{z_{IL}}_{X_{I_k}^G, T_L^I}$ are used for state and covariance update. Here $\ominus$ operator denotes difference between quantities lying on-manifold.

5.4.3.5 Camera - LiDAR Update

Camera-LiDAR update under EKF framework for performing LiDAR-IMU-Camera joint calibration is the novel contribution of this thesis. The measurement residual in this case involves alignment of calibration target plane parameters $[n_L, d_L]_i$, $[n_C, d_C]_j$ detected in LiDAR (using the

Point Cloud Library [51]) and camera (using OpenCV[49]) sensors, at times i and j respectively. As far as LiDAR is considered, as shown in Figure 5.4, the calibration target detection is performed on ego motion compensated scan. The measurement residuals are as given:

$$r_{CL_{ij}} = \begin{bmatrix} r_R \\ r_p \end{bmatrix} = \begin{bmatrix} n_C - {R_C^I}^\top R_G^{Ij} {R_G^{Ii}}^\top R_L^I n_L \\ n_C^\top p_{L_i}^{C_j} + d_C - d_L \end{bmatrix} \tag{5.12}$$

Where $p_{L_i}^{C_j} = {R_C^I}^\top R_G^{I_j}({R_G^{I_i}}^\top p_L^I + p_{I_i}^G) - {R_C^I}^\top (p_C^I + R_G^{I_j} p_{I_j}^G)$. The residual $r_{CL_{ij}}$ and the corresponding Jacobians $H^{z_{ij}}_{T_{I_i}^G, T_{I_j}^G, T_L^I, T_C^I}$ which are functions of state variables $T_{I_i}^G$, $T_{I_j}^G$, T_L^I, T_C^I, are used for state and covariance update. As the residual (Equation A.37) is a function of both T_L^I and T_C^I, this EKF update step jointly updates/estimates Camera-IMU and LiDAR-IMU extrinsic calibration parameters. While this residual has been used for pair-wise calibration of Camera-LiDAR systems([24],*surface point to plane constraint* described in Chapter 2 Section 2.4.3.1), it has been implemented here to perform joint calibration of Camera, IMU and LiDAR together. The measurement Jacobians (Equation 5.13) are evaluated at the best available estimate of the state variables $x = \{T_{I_i}^G, T_{I_j}^G, T_L^I, T_C^I\}$.

$$\begin{aligned}
\mathbf{H}^{z_{ij}}_{T_{I_i}^G} &= -\frac{\partial r_{CL_{ij}}}{\partial T_{I_i}^G}\bigg|_{\hat{x}=\{\hat{T}_{I_i}^G, \hat{T}_{I_j}^G, \hat{T}_L^I, \hat{T}_C^I\}} \\
\mathbf{H}^{z_{ij}}_{T_{I_j}^G} &= -\frac{\partial r_{CL_{ij}}}{\partial T_{I_i}^G}\bigg|_{\hat{x}=\{\hat{T}_{I_i}^G, \hat{T}_{I_j}^G, \hat{T}_L^I, \hat{T}_C^I\}} \\
\mathbf{H}^{z_{ij}}_{T_C^I} &= -\frac{\partial r_{CL_{ij}}}{\partial T_C^I}\bigg|_{\hat{x}=\{\hat{T}_{I_i}^G, \hat{T}_{I_j}^G, \hat{T}_L^I, \hat{T}_C^I\}} \\
\mathbf{H}^{z_{ij}}_{T_L^I} &= -\frac{\partial r_{CL_{ij}}}{\partial T_L^I}\bigg|_{\hat{x}=\{\hat{T}_{I_i}^G, \hat{T}_{I_j}^G, \hat{T}_L^I, \hat{T}_L^I\}}
\end{aligned} \tag{5.13}$$

The details concerning Jacobian calculation can be found in Chapter A Sections A.1 & A.4.

5.4.3.6 Update Equations

Depending on the type of measurement update (Sections 5.4.3.3 or 5.4.3.4 or 5.4.3.5), the individual Jacobians are stacked together to form a consolidated Jacobian $\mathbf{H}_k$. Along with the

respective measurement residual $\mathbf{r}_k$, the consolidated Jacobian $\mathbf{H}_k$ are used for state and state-covariance update when a measurement update is available. The update equations are presented in Equations 5.14-5.16.

$$\mathbf{K}_k = \mathbf{P}_{k\_}\mathbf{H}_k^\top(\mathbf{H}_k\mathbf{P}_{k\_}\mathbf{H}_k^\top + \mathbf{R})^{-1} \tag{5.14}$$

$$\begin{bmatrix} \hat{X}^G_{I_{k+}} \\ \hat{T}^I_{C+} \end{bmatrix} = \begin{bmatrix} \hat{X}^G_{I_{k-}} \\ \hat{T}^I_{C-} \end{bmatrix} \oplus \mathbf{K}_k\mathbf{r}_k \tag{5.15}$$

$$\mathbf{P}_{k+} = \mathbf{P}_{k\_} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k\_} \tag{5.16}$$

Here $\mathbf{K}_k$ is the Kalman gain which is used in Equations 5.15 and 5.16 for state and state covariance update respectively. $\mathbf{R}$ is the tunable measurement covariance matrix. '-' denotes the estimate before update while '+' denotes the estimate after update. $\oplus$ in Equation 5.15 refers to generic composition which can be algebraic addition for variables on vector space or rotation composition for variables on $SO(3)$.

5.5 System Description

Experiments were performed with data from the sensor suite (Figure 5.1) consisting an Ouster 128 Channel LiDAR running at 10 Hz, a Basler Ace Camera $[1600 \times 1200]$ running at 30 Hz, and a Vectornav VN-300 Camera running at 400 Hz. Calibration data is collected from these sensors while exciting the sensor suite along all directions and about all axes.

5.6 Experiments

5.6.1 Comparison of jointly estimated T_C^I with individually estimated T_C^I

In the absence of ground truth, the estimation of T_C^I from the proposed joint calibration framework is compared against T_C^I estimated using Kalibr[1][7] Camera-IMU calibration tool-box, and also with the author's own implementation of Camera-IMU calibration method[2] published previously as [8] and discussed in detail in Chapter 4. The results have been tabulated in Table 5.1. The

[1] `https://github.com/ethz-asl/kalibr`
[2] `https://github.com/unmannedlab/camera_imu_calibration`

Dataset	Method	R°	P°	Y°	x [cm]	y [cm]	z [cm]
1	**Kalibr [7]**	88.9694	1.0825	90.8270	7.33	12.58	-7.47
	Individual Calibration (IC) [8]	88.9471	1.0832	90.9106	7.26	12.49	-7.77
	Proposed Joint Calibration (JC)	89.0615	1.0460	90.7545	7.32	12.43	-7.96
	Difference (IC $\ominus$ JC)	-0.1174	0.1566	-0.0346	-0.06	0.08	0.22
	|% change (IC $\ominus$ JC)|	0.1320	14.4607	0.0381	0.8796	0.6605	2.7825
2	**Kalibr [7]**	88.9854	1.0673	90.8232	7.33	12.67	-7.54
	Individual Calibration (IC) [8]	89.0770	1.0407	90.7982	7.21	12.70	-8.05
	Proposed Joint Calibration (JC)	89.2415	1.0122	90.6153	7.28	12.61	-8.18
	Difference (IC $\ominus$ JC)	-0.1678	0.1832	-0.0261	-0.08	0.12	0.16
	|% change (IC $\ominus$ JC)|	0.1884	17.6055	0.0287	1.0475	0.9242	2.0032
3	**Kalibr [7]**	89.0208	1.0706	90.7984	7.33	12.59	-7.54
	Individual Calibration (IC) [8]	88.9734	1.0444	90.8052	7.15	12.52	-8.11
	Proposed Joint Calibration (JC)	89.2148	1.0413	90.8143	7.10	12.29	-7.79
	Difference (IC $\ominus$ JC)	-0.2412	-0.0091	-0.0032	-0.0037	0.26	-0.31
	|% change (IC $\ominus$ JC)|	0.2711	0.8675	0.0036	0.0511	2.0799	3.8819
4	**Kalibr [7]**	89.0547	1.0927	90.8185	7.27	12.54	-7.56
	Individual Calibration (IC) [8]	89.1348	1.0712	90.7966	6.98	12.35	-7.99
	Proposed Joint Calibration (JC)	89.2269	1.0738	90.7073	6.96	12.28	-8.06
	Difference (IC $\ominus$ JC)	-0.0939	0.0892	0.0038	0.01	0.09	0.07
	|% change (IC $\ominus$ JC)|	0.1053	8.3252	0.0042	0.2001	0.7015	0.9297
5	**Kalibr [7]**	89.0552	1.0770	90.8315	7.30	12.56	-7.51
	Individual Calibration (IC) [8]	89.0694	1.0830	90.9919	6.79	12.53	-7.71
	Proposed Joint Calibration (JC)	89.2991	1.0970	90.7897	6.86	12.30	-7.95
	Difference (IC $\ominus$ JC)	-0.2335	0.2019	0.0165	-0.09	0.25	0.26
	|% change (IC $\ominus$ JC)|	0.2621	18.6446	0.0181	1.2720	2.0075	3.3742
σ (Standard Deviation)	**Kalibr + IC + JC**	0.1106	0.0236	0.0830	0.1839	0.1367	0.2460

Table 5.1: **Estimation of T_C^I:** T_C^I estimated using the open source camera-IMU calibration tool box Kalibr [7], our own implementation of [8] and the proposed method of joint calibration. Kalibr[7] may be considered as the reference to compare our results.

Camera-IMU calibration T_C^I estimation from the proposed joint calibration approach is very close to the individual calibration from Kalibr [7] $\&$ our own implementation of [8], as evident from low standard devitation of 0.1106°, 0.0236°, 0.0830° along roll, pitch $\&$ yaw and 0.1839 cm, 0.1367 cm, 0.2460 cm along x, y $\&$ z axes. As presented in Table 5.1 $\&$ Table 5.3, the difference and the $\%$ change between individually and jointly estimated T_C^I is, in general, very low.

5.6.2 Comparison of jointly estimated T_L^I with individually estimated T_L^I

Unlike Camera IMU calibration where Kalibr[7] has been used as an independent source for comparison and reference, there is no independent reference to compare LiDAR-IMU calibration. In this section the difference between individually and jointly estimated T_L^I has been presented in

Dataset	Method	R°	P°	Y°	x [cm]	y [cm]	z [cm]
1	**Individual Calibration**	1.2942	0.8770	179.2007	0.93	-13.15	-9.28
	Proposed Joint Calibration	0.3288	1.7099	179.7766	2.25	-12.36	-6.16
	Difference	0.9825	-0.8362	-0.5711	-1.61	-0.77	-3.28
	|% **change (IC** $\ominus$ **JC)**|	298.7820	48.9010	0.3177	71.7457	6.2071	53.3309
2	**Individual Calibration**	1.7852	0.0695	178.9726	0.49	-12.85	-8.63
	Proposed Joint Calibration	0.5692	1.5421	179.7952	2.67	-12.55	-7.34
	Difference	1.2380	-1.4806	-0.8083	-2.64	-0.27	-1.54
	|% **change (IC** $\ominus$ **JC)**|	69.3	2131.7	0.5	543.4	2.1	17.9
3	**Individual Calibration**	0.3885	-0.0703	179.0941	0.12	-12.91	-7.40
	Proposed Joint Calibration	0.6141	1.6429	-179.9208	2.26	-13.29	-6.27
	Difference	-0.1976	-1.7234	-0.9671	-2.28	0.49	-1.43
	|% **change (IC** $\ominus$ **JC)**|	50.9	2453.1	0.5	1926.1	3.8	19.3
4	**Individual Calibration**	0.6957	0.3212	179.5279	0.39	-13.51	-7.56
	Proposed Joint Calibration	0.5177	1.4934	179.9749	2.00	-14.12	-6.84
	Difference	0.1896	-1.1762	-0.4366	-1.80	0.65	-0.87
	|% **change (IC** $\ominus$ **JC)**|	27.2478	366.2128	0.2432	466.0055	4.8106	11.4578
5	**Individual Calibration**	0.1304	0.3980	178.9583	-0.36	-12.60	-7.55
	Proposed Joint Calibration	0.7718	1.7005	-179.6015	2.23	-14.44	-6.57
	Difference	-0.5986	-1.3212	-1.4229	-2.58	2.02	-1.39
	|% **change (IC** $\ominus$ **JC)**|	458.9265	331.9219	0.7951	711.5273	15.9964	18.4284

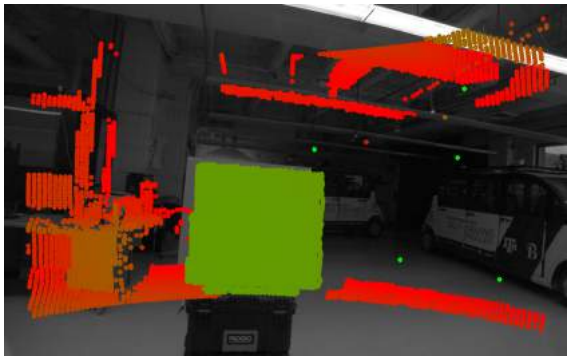
Table 5.2: **Estimation of T_L^I:** Comparing Individual Calibration with Joint Calibration.

Table 5.2. The difference and hence the percentage change (Tables 5.2 & 5.3) between independently and jointly estimated LiDAR-IMU T_L^I calibration estimate is larger than the difference and hence the percentage change in Camera-IMU T_C^I calibration values (Tables 5.1 & 5.3).

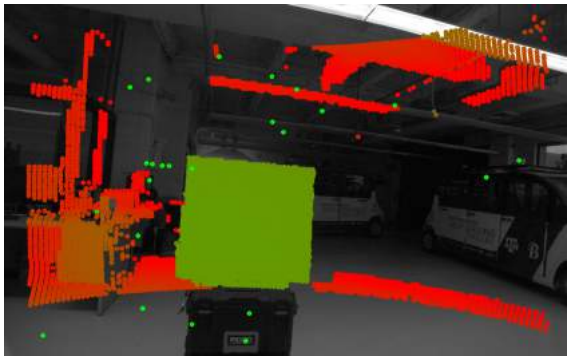
5.6.3 Projection of LiDAR points on Camera by daisy chaining estimated T_C^I & T_L^I

Although T_C^I from various methods in Table 5.1 look very similar across individual and joint calibration, the projection of LiDAR points ($\{X_L\}$) on image plane look different with individually (self implementation of [8] for camera-IMU, Chapter 3 for LiDAR-IMU) and jointly estimated T_C^I & T_L^I [3] in Figures 5.5a, 5.5c, 5.5e and 5.5b, 5.5d, 5.5f respectively. This implies that the difference in T_L^I between individual and joint calibration, tabulated in Tables 5.2 & 5.3, is the primary cause for the difference in how the projection of LiDAR points look (Figure 5.5). When $\{X_L\}$ are projected onto the image plane using individually estimated T_C^I & T_L^I, the projected points do not align well with the corresponding image pixels. As visible in Figures 5.5a, 5.5c, 5.5e, the projected

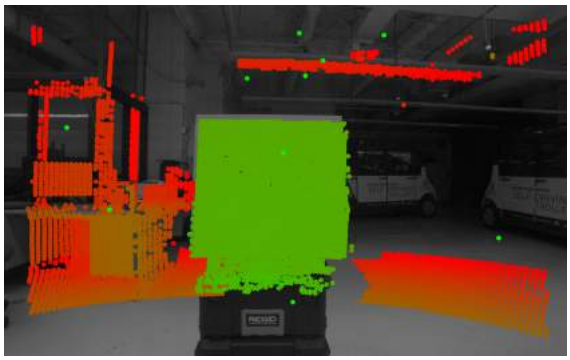
[3] $T_L^C = {T_C^I}^{-1} T_L^I$, $X_C = T_L^C X_L$



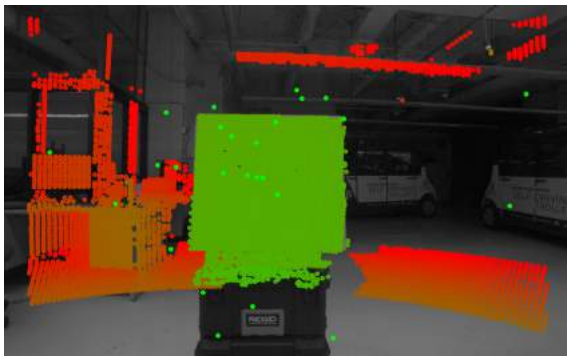
(a) **Individual Calibration:** Projected LiDAR points are misaligned.



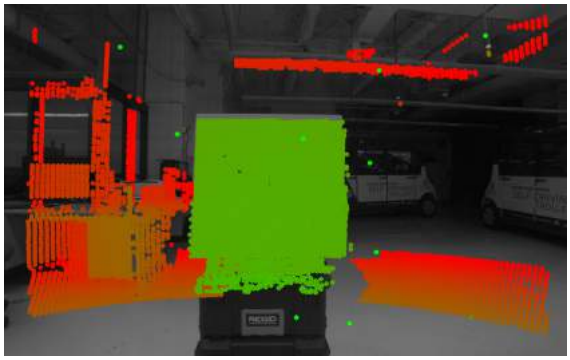
(b) **Joint Calibration:** Reduction in misalignment of projected LiDAR points.



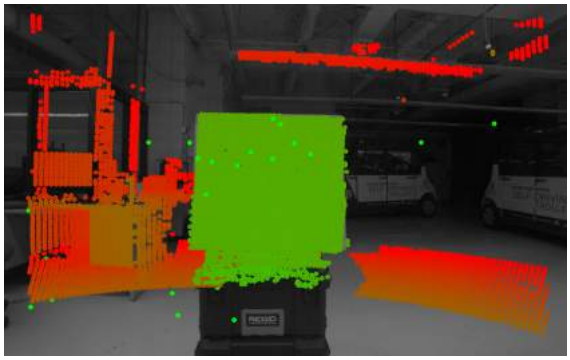
(c) **Individual Calibration:** Projected LiDAR points are misaligned.



(d) **Joint Calibration:** Reduction in misalignment of projected LiDAR points.



(e) **Individual Calibration:** Projected LiDAR points are misaligned.



(f) **Joint Calibration:** Reduction in misalignment of projected LiDAR points.

Figure 5.5: **Qualitative Evaluation of Individual vs Joint Calibration**: Daisy chaining estimated T_L^I & T_C^I for projecting LiDAR points on Camera Image.

% Change in values of $\mathbf{T_C^I}$ & $\mathbf{T_C^I}$ from Individual Calibration to Joint Calibration							
Dataset	% change in	**Roll**	**Pitch**	**Yaw**	**x**	**y**	**z**
1	$\mathbf{T_C^I}$	0.1320	14.4607	0.0381	0.8796	0.6605	2.7825
	$\mathbf{T_L^I}$	298.7820	48.9010	0.3177	71.7457	6.2071	53.3309
2	$\mathbf{T_C^I}$	0.1884	17.6055	0.0287	1.0475	0.9242	2.0032
	$\mathbf{T_L^I}$	69.3	2131.7	0.5	543.4	2.1	17.9
3	$\mathbf{T_C^I}$	0.2711	0.8675	0.0036	0.0511	2.0799	3.8819
	$\mathbf{T_L^I}$	50.9	2453.1	0.5	1926.1	3.8	19.3
4	$\mathbf{T_C^I}$	0.1053	8.3252	0.0042	0.2001	0.7015	0.9297
	$\mathbf{T_L^I}$	27.2478	366.2128	0.2432	466.0055	4.8106	11.4578
5	$\mathbf{T_C^I}$	0.2621	18.6446	0.0181	1.2720	2.0075	3.3742
	$\mathbf{T_L^I}$	458.9265	331.9219	0.7951	711.5273	15.9964	18.4284

Table 5.3: % Change between individually calibrated $\mathbf{T_C^I}$, $\mathbf{T_L^I}$ and jointly calibrated $\mathbf{T_C^I}$, $\mathbf{T_L^I}$ respectively. The change in T_L^I estimates is larger.

calibration board 3D points (green) do not align with the corresponding checkerboard image pixels (that belong to the checkboard calibration target), whereas the mis-alignment is significantly reduced in Figures 5.5b, 5.5d, 5.5f - where $\{X_L\}$ have been projected using jointly estimated T_C^I & T_L^I. The misalignment in projections visible in Figures 5.5a, 5.5c, 5.5e could be due to incorrect T_C^I and/or T_L^I. However, as presented in Table 5.3 the difference between individually and jointly estimated T_C^I is smaller than the difference between individually and jointly estimated T_L^I, which may lead one to conclude that individually estimated T_L^I is more error prone than individually estimated T_C^I.

5.7 Conclusion

This Chapter presented a novel technique to jointly calibrate a system of LiDAR, IMU and Camera under the EKF framework which the author believes is the only method contributed to the existing literature as far as joint calibration of a LiDAR-IMU-Camera system is considered. The experiments in Section 5.6.3 (specifically Figure 5.5) show that joint calibration of sensors yielded better results than the case when the sensors were individually calibrated. Moreover, the change in the LiDAR-IMU calibration estimate is more pronounced than the change in camera-IMU calibration estimate (Tables 5.2 & 5.3) which implies that pair-wise calibration of LiDAR-

IMU systems is more prone to errors. Besides, as explained in Chapter 3, Section 3.8.5, it is difficult to verify LiDAR IMU systems without using auxiliary sensors. So, a natural conclusion is that LiDAR-IMU systems not only need an auxiliary sensor, preferably a visual sensor like camera, for verification of extrinsic calibration but also require joint calibration with the auxiliary sensor in-order to improve the LiDAR-IMU calibration estimate.

As discussed in Section 1.4 of Chapter 1, one of the goals of Part I was to combine methods of pairwise calibration of LiDAR, IMU & Camera to devise an approach to jointly register them all together, which has been demonstrated in the current Chapter.

There are some immediate areas of future work as far as the sytem in the current Chapter is considered. First, in addition to aligning the plane of the calibration target detected in both LiDAR and Camera, the edges of the calibration target should detected and aligned, as presented in Chapter 2 on Camera-LiDAR pair-wise calibration. This will further improve the result of joint estimation and also provide a method for quantifying the result of calibration using Mean Line Re-projection Error (MLRE) (Chapter 2, Section 2.7.3). Secondly, given that a calibration board already constrains the estimation problem it should be explored both mathematically and experimentally if it is still necessary to motion excite the sensor suite along all degrees of freedom for ensuring full observability of the registration parameters. Lastly, in addition to performing extrinsic calibration the framework must also include temporal calibration of the concerned sensors.

As far as the broader context of multi-sensor fusion and state estimation is concerned, the proposed approach presents how sensor measurements from sensors of very different modalities can be used to determine the static registration between each sensor pair in the system and also the evolving pose and velocity of the system with respect to a frame of reference. The method can be easily extended to N different sensors of M different types and can be used to determine N-1 static registration parameters and also the evolving pose of the sensor system. The quality of registration however depends on the quality of data collected for the purpose.

Ideally, before the sensor suite is deployed on a physical system for autonomous operation, the objective should be to obtain the best estimates of the static registration parameters between pairs of

sensors, that is to say the objective is to prioritize extrinsic calibration of sensor pairs. To fulfill this objective the best quality of calibration data must be provided by adequately motion exciting the sensor suite along all degrees of freedom with the calibration target in the common field of view of the concerned sensors[4]. Once extrinsic calibration is achieved, the sensor suite can be installed on a robot and then the same multi-sensor pose estimation and mapping framework that was used for determining extrinsic calibration must be used once again (prioritizing robot pose estimation this time) for estimating the evolving pose of the robot with the added ability to continously monitor the extrinsic calibration estimates during the course of the robot's operation. Future work should explore how to overcome difficulties in calibration estimate observability during normal operating conditions of the robot on which the mobile sensor suite is mounted.

This Chapter concludes Part I which dealt with registration of sensors on a mobile sensor suite with three very commonly used sensors, *viz.* Cameras, IMU and LiDAR. Part II of the thesis deals with methods of registration of static infrastructure sensors spread over large areas which may or may not share fields of view. Unlike Part I where the handy sensor suite is intended to be mounted on a mobile robot, the infrastructure sensors in Part II will not be mounted on a robot but instead on static infrastructure like parking lots, factory, warehouses, complex traffic intersections, etc. such that they can augment an autonomous vehicle's perception and state estimation abilities by relaying information they sense to the AV over V2X technology, and also by aiding the AV's localization and mapping software stack. The limitations while attempting to perform registration or extrinsic calibration of static infrastructure sensors is that they may not share fields of view with other sensors hence making usage of calibration targets difficult, they may be located at large heights making detection of calibration markers difficult, they cannot be motion excited (unlike the sensor suite in Part I).

Part II covers two important infrastructure sensor registration problems in Chapters 6 & 7 respectively. The first concerns registration of a large network of static infrastructure cameras in an indoor facility using a calibration robot. Such a system can be used for automated parking, robot

[4]If feature association across sensors is solved then using a calibration target may not be necessary

localization in controlled settings like factory floor and warehouses, and also for security surveillance in airports, shopping malls etc. Such a distributed system of infrastructure sensors need to be registered so that they can aid autonomous systems and perform automated surveilance and monitoring. The registration is performed by solving a large multi-variable smoothing optimization problem where the static poses of the infrastructure sensors and the evolving pose of the mobile calibration robot are all estimated together in a fixed reference frame. Among other applications, the framework can be easily extended to do multiple robot localization for robots operating under the survelliance of the system of static infrastructure sensors. The second registration problem concerns localization of static downward looking fisheye camera in a prior map. Such cameras will be mounted on complex traffic intersections to aid an AV by relaying it information about areas which lie in the AV's blind spots and also by helping in situations in which on board sensors malfunctions or suffer from limited field of view. For such systems to be beneficial they need to registered in the same map as the AV. The registration is done using maximization of mutual information between the acquired fisheye image and prior mapping data comprising satellite and LiDAR maps. This method takes a departure from sparse methods used so far to use entire appearance information to register sensors of different sensing modalities.

PART II - REGISTRATION OF STATIC SMART INFRASTRUCTURE SENSORS

6. EXTRINSIC CALIBRATION OF LARGE NETWORK OF INFRASTRUCTURE CAMERAS

6.1 Introduction

Until now, the thesis discussed methods to register sensors rigidly attached to a handy sensor suite that can be easily mounted on robots like self driving vehicles, unmanned ground vehicles and autonomous aerial robots. The registration method can not only be used to determine the static extrinsic calibration between sensor pairs but also used for estimating the evolving pose of the sensor suite. In Part II the thesis discusses methods to register sensors rigidly attached to infrastrucuture like parking lot, traffic posts at intersections, etc. In this Chapter the thesis presents a batch optimization based technique to perform registration of several cameras. A mobile calibration robot with a fisheye camera is used to determine the static registration of fixed infrastructure sensors in addition to estimating its own pose by tracking environment features and the infrastructure cameras in the field of view. The two way detection of infrastructure cameras in the calibration robot and vice versa constrains the global optimization both ways. The method demonstrates how data from sensors both static and moving can be used to determine their own poses with respect to a fixed frame. The approach can be extended to perform multi-sensor calibration, localization and

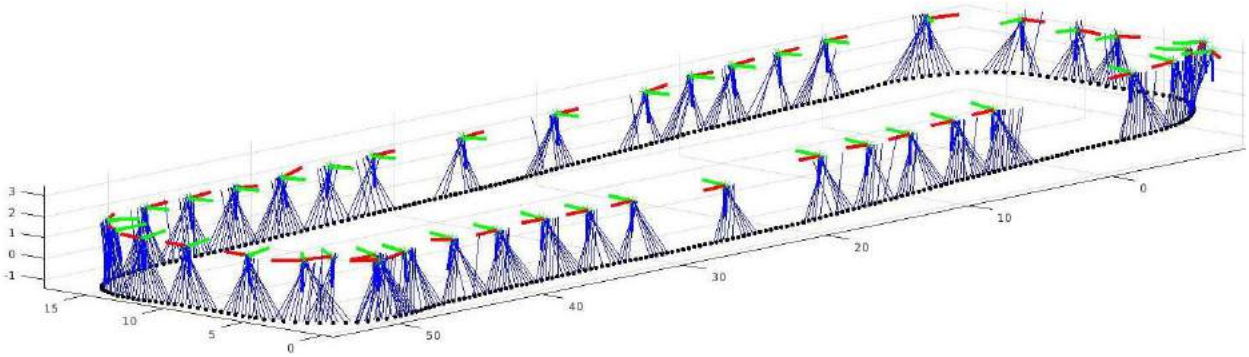


Figure 6.1: **Graph of optimized edge nodes and robot trajectory**: The estimated infrastructure camera poses (as coordinate frames) and the robot trajectory (as dark dotted lines). The blue straight lines are robot-mounted aruco marker detections from the infrastructure cameras.

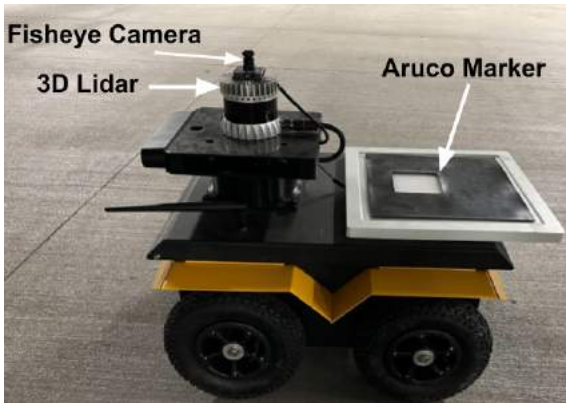
mapping in areas surveilled by several infrastructure cameras.

6.2 Motivation & Overview

This work presents automated extrinsic calibration of a large number of fixed, ceiling-mounted infrastructure cameras using a mobile robot equipped with a single upward-facing fisheye camera and a backlit marker for easy detection. The fisheye camera is used to perform visual odometry, while at the same time the marker on the robot is seen by the infrastructure camera. The fisheye camera is also able to detect the infrastructure cameras. This two-way, bidirectional detection constrains the pose of the robot as well as of all the infrastructure cameras to a common coordinate frame by solving a global optimization problem. Such an approach may be used to automatically register a large-scale multi-camera system used for surveillance, automated parking, or robotic applications in controlled environments like warehouses, factory floor, etc. The proposed method has been extensively validated using real-world experiments and and comparisons against a LiDAR based approach has been provided.

Infrastructure based sensing is gaining popularity with the goal to equip the environment of an autonomous system with sensors that can eliminate the dependence on onboard sensors for perception and localization to operate in controlled environments like parking lots, warehouses, and the factory floor. A distributed sensing system in a parking lot can autonomously park vehicles with drive-by-wire technology and direct the movement of cars through the environment. However, for such a network of fixed infrastructure cameras to be useful to an autonomous robot for any of the aforementioned applications, one needs to know their extrinsic calibration - the 6 Degree of Freedom (DoF) pose of each infrastructure camera relative to a common coordinate frame known to the cameras and the robot. A tedious way to determine the camera poses would involve placing a fiducial marker (Aruco/Checkerboard) [66] in the common FoV between every pair of infrastructure cameras, getting their relative poses and subsequently daisy-chaining the poses through the sequence of cameras and back to the starting camera. The manual method of registering multiple sensors requires overlapping field of view (FoV) and tiresome manual labor every time the system needs to be re-calibrated.

(a) **Edge nodes:** Experiments were performed with a distributed sensing system of $\sim$ 40 edge devices mounted throughout the ceiling of our test environment. Each edge node comprises of an RGB camera (called as infrastructure camera in the paper) and compute. These are powered over ethernet and networked and time-synced to a central server.



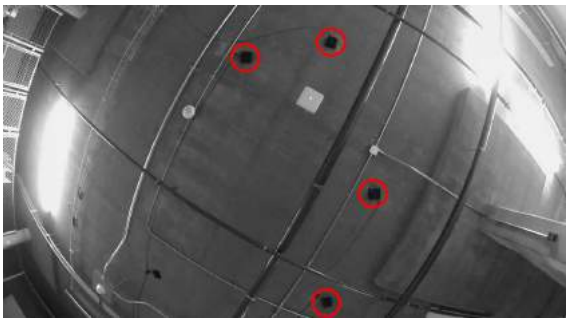
(b) **Calibration Robot:** Clearpath Jackal with an upward-facing wide-angle fisheye camera F and an aruco marker M. The aruco marker is backlit to make detection easier. The robot also carries a LiDAR for comparison between LiDAR and camera-only calibration.

Figure 6.2: Edge nodes and calibration robot

The proposed work presents an approach for solving the extrinsic calibration problem for a large network of distributed cameras with no manual intervention. Experiments have been performed with a network of $\sim$40 infrastructure cameras mounted on the ceiling of an indoor facility (Figure 6.2a), along with a mobile calibration robot (Figure 6.2b) mounted with an upward-facing fisheye camera that traverses through that facility. The robot also carries a backlit Aruco marker making its detection in infrastructure cameras easier. Detections/measurements are taken in both directions: upwards from the robot to the infrastructure camera (Figure 6.3a) and also, from each of the infrastructure camera down onto the robot (as it passes through its FoV, Figure 6.3b), along with visual odometry (VO) [67] from the robot to jointly optimize for the poses of the robot and each infrastructure camera relative to a common coordinate frame.

6.3 Literature Review

The automatic identification of the relationship between multiple fixed cameras in a network has been done in terms of the network topology by [68] [69]. However, this is used for the detection



(a) **Looking up ↑ Edge-Node detection from Robot:** Automated detection of ceiling-mounted infrastructure cameras when viewed from the upward-facing fisheye camera on the calibration robot.

(b) **Looking down ↓ Robot Detection from Edge Node:** The Calibration Robot viewed from an infrastructure camera with axes drawn on the detected backlit Aruco marker.

Figure 6.3: **Looking both ways:** Looking up from the calibration robot to the ceiling, looking down from the ceiling to the calibration robot.

and re-id of people being tracked through these cameras. One of the first works to describe the joint calibration of a system of surveillance cameras and a mobile robot [70] used homography between the image plane of each camera and the ground plane to calibrate the two. As the robot moved around the FoV of each camera, its simultaneous detection in the camera image plane and localization using LiDAR on the ground plane allowed the automated computation of homography between each camera and the ground plane. More recently, Zou et. al. [71] describe a set of fisheye cameras deployed at an intersection to monitor traffic. Each camera-node (comprising of sensor and compute) is extrinsically calibrated using homography between hand-picked points on the road (visible in the camera image) that are also discernible in a satellite image of the area. [72] describes an extrinsic calibration method for RGB-D camera networks for non overlapping static and dynamic cameras where the authors perform visual odometry based calibration and depth image based calibration refinement. [72] first does extrinsic calibration of overlapping cameras using several fiducial tags [29] and then bridges the non overlapping views through visual odometry [67], and in the end they refine the calibration result by using the depth images of the RGB-D cameras by detection and alignment of the floor plane. All of the above approaches to register

multiple cameras (RGB & RGB-D) cater only to small environments ($\sim$1 - 2 living rooms) and small number of cameras ($\sim$ 2 - 5 cameras).

6.4 System Description & Notation

This section describes the various components of the overall system and introduces the notation used in the work.

6.4.1 The Distributed Camera System

The distributed camera system (Figures 6.2a & 6.3a) comprises several static infrastructure cameras which may or may not have overlapping fields of view. They generate images of size 1080 x 1920 pixels at 30 Hz. The camera forms a component of the edge device shown in Figure 6.2a. In addition to RGB sensing the node can also measure depth, and has onboard computing capabilities. In this work the focus is to just use the RGB camera sensor for registering the multi camera system.

6.4.2 The Calibration Robot

A Clearpath Jackal (Figure 6.2b) is used as the calibration robot . The robot has a rigidly attached upward-facing wide-angle (160°) 2MP fisheye camera which acquires images of size 1080 x 1920 pixels at 30 Hz. The fisheye camera on the robot is kept facing upward because the ceiling (height varying between 3 - 4 m from floor level) of the indoor facility ensures the presence of sufficient static features to track for robot pose estimation using VO. A wide-angle lens is being used for VO in-order to cover a large FoV and track several features. Moreover, a wide-angle camera ensures that several infrastructure cameras can be detected (using OpenCV [49] blob detection, Figure 6.3a) and can be used to further constrain the position of the detected cameras in the estimation procedure. The fisheye camera calibration toolbox available in MATLAB [13] is used to obtain the fisheye camera's intrinsics. A backlit aruco marker [66] of known dimensions is attached to the calibration robot so that the robot can be easily detected by the infrastructure cameras (Figure 6.3b). The calibration robot also has a 64 Channel Ouster 3D-LiDAR which generates 3D scans at 10 Hz and can be used for LiDAR Odometry (LO) [73] based calibration

of infrastructure cameras - a method used for benchmarking and comparison against the proposed VO based approach.

6.4.3 Notation

In this work, 6-DoF pose is expressed as a homogeneous transformation matrix $T = \begin{bmatrix} R(q) & p \\ 0 & 1 \end{bmatrix}$, where rotation R is represented as quaternion q and position p as a $R^{3\times1}$ vector. The global frame W coincides with the origin of the robot motion estimated using VO. The pose T_F^W of the fisheye camera F is synonymous with the calibration robot pose. There are N cameras $\{C_k\}_{k=0:N-1}$ in the distributed infrastructure camera system with corresponding global poses $\{T_{C_k}^W\}_{k=0:N-1}$. The detection of the aruco marker M on the calibration robot in an infrastructure camera C_k results in aruco detection pose measurement $T_M^{C_k}$ which represents the pose of the aruco marker in camera C_k frame. The fisheye camera F and the aruco marker M have a spatial offset T_M^F between them. The fisheye camera F detects and tracks point features (as pixels $\in R^2$) to do VO for robot pose estimation. The fisheye camera also detects infrastructure cameras $\{C_k\}_{k=0:N-1}$ as 2-D pixel ($\in R^2$) measurements on its image plane which are ultimately used to constrain all variables optimized in the estimation process.

6.4.4 Time Synchronization

All edge nodes are synchronized using an NTP (Network Time Protocol) server. A central server is set up as the NTP server and all the edge nodes and the calibration robot use this server to sync their time as NTP clients. The resulting synchronization has an error in the range of 1 to 2 milliseconds on the client devices in relation to the time on the server. However, since the sensors themselves (cameras and LiDAR on the robot vs the infrastructure cameras) are asynchronous (not triggered together and have different frame rates), there is a small variable error between images and scans captured.

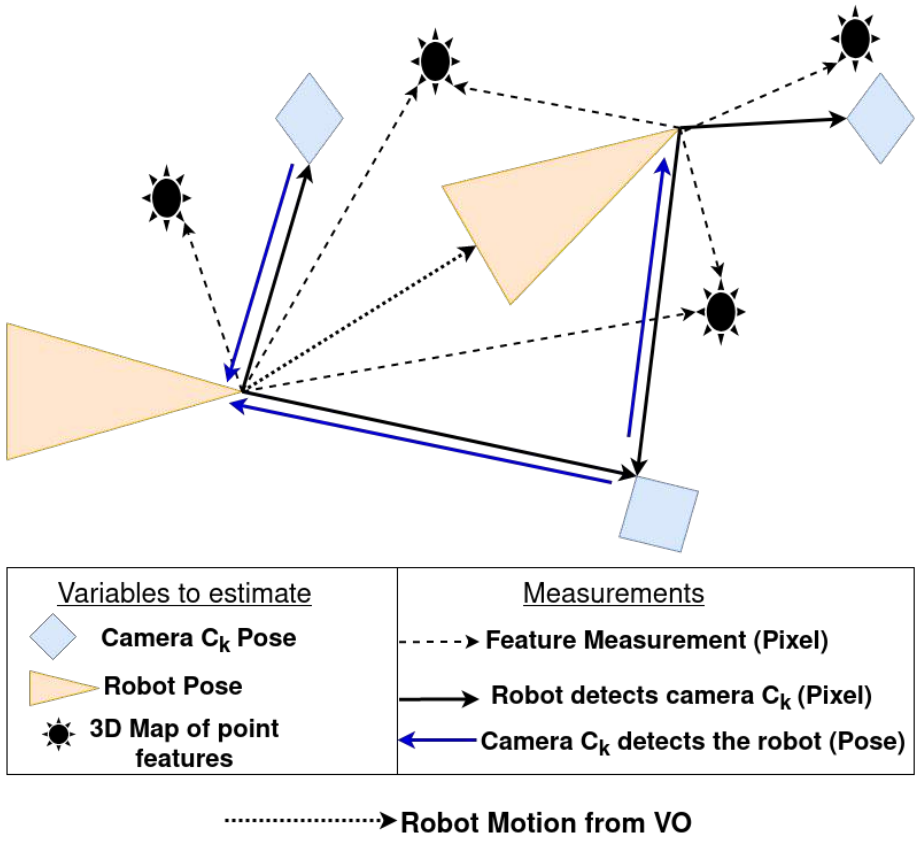


Figure 6.4: **Multi-sensor registration**: Schematic shows the variables estimated using the given measurements. In practice, an infrastructure camera C_k detects the rigidly attached aruco marker M on the robot.

6.5 Problem Statement

An overview of the problem to be solved with the known measurements and unknown variables to be estimated is given in Figure 6.4. The primary goal of this work is to determine the pose $\{T^W_{C_k}\}_{k=0:N-1}$ of infrastructure cameras $\{C_k\}_{k=0:N-1}$ (Figure 6.3a) with respect to a global frame W. To this end, one needs to estimate the calibration robot pose T^W_F and the constant spatial offset T^F_M between the fisheye camera F and the aruco marker M such that an aruco detection pose measurement like $T^{C_k}_M$ can be used to estimate $T^W_{C_k}$ - camera C_k's global pose (since $T^W_{C_k} = T^W_F T^F_M (T^{C_k}_M)^{-1}$). VO is used to determine the unscaled robot pose T^W_F (and unscaled map $\{X_k \in R^3\}_{k=0:P-1}$ of 3D points from 2D feature detection and tracking) (Section 6.6.1). Next, the aruco pose measurements $T^{C_k}_M$ are used to: 1. determine the metric scale s of the robot pose T^W_F (Section 6.6.2), & 2. estimate T^F_M (Section 6.6.3). Finally, two way measurements i.e. aruco detection pose measurements from detection of aruco marker in the infrastructure cameras $\{C_k\}_{k=0:N-1}$ and the detection of the infrastructure cameras $\{C_k\}_{k=0:N-1}$ in the fisheye camera F are used to jointly optimize the infrastructure camera poses $\{T^W_{C_k}\}_{k=0:N-1}$ along with other variables like the robot pose T^W_F and the environment map $\{X_k \in R^3\}_{k=0:P-1}$ of 3D points (Section 6.7).

6.6 Method

A 4 stepped approach is used to solve this problem, they are 1. Estimate un-scaled robot motion with the upward-looking fisheye camera F using Visual Odometry (VO) (Section 6.6.1), 2. Determine the scale of VO using aruco detections (Section 6.6.2), 3. Determine the spatial offset between the aruco marker and the upward-looking fisheye camera (Section 6.6.3), 4. Estimate the poses of the distributed infrastructure cameras (Section 6.6.4).

6.6.1 Estimate Motion of the upward-looking fisheye camera using Visual Odometry (VO)

Semi-direct Visual Odometry (SVO) [3] is used to determine the motion of the upward-looking fisheye camera on the robot. SVO is chosen over other VO algorithms because of its ease of use and proven ability with wide-angle lenses [74]. Additionally, SVO can be optimized to work for downward-looking cameras in aerial robots where the camera tracks features on a dominantly

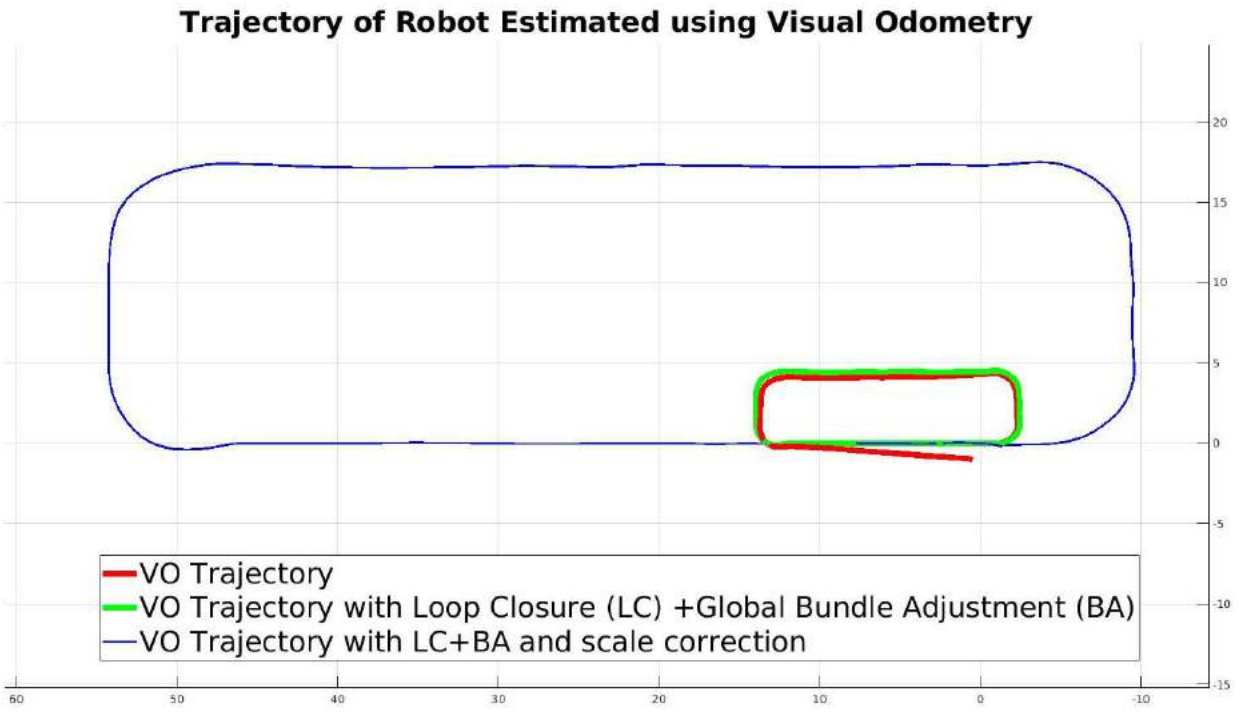


Figure 6.5: **Motion estimation using Visual Odometry (VO).** The trajectory in red is the result from SVO [3]. Loop closure [4] and bundle adjustment [5] are used to reduce the drift in VO (shown in green) and finally the metric scale (in meters) (Sections 6.6.2 & 6.6.3) of the trajectory is determined using the detection of aruco marker on the calibration robot.

flat surface, a situation which is similar to the present context where an upward-looking camera tracks static features on the ceiling. Drift in VO is eliminated by implementing loop closure (using DBoW2 [4] for place recognition) and global bundle adjustment (in ceres solver [5]) using the reprojection error residual Equation 6.1 for the minimization problem in Equation 6.2. (Figure 6.5).

$$r_{kj} = \begin{bmatrix} u \\ v \end{bmatrix}_{kj} - \pi(K, T_{F_j}^W, X_k) \tag{6.1}$$

$$\{\hat{T}_{F_j}^W\}_{j=0:M-1}, \{\hat{X}_k\}_{k=0:P-1} = \underset{\{T_{F_j}^W\}_{j=0:M-1}, \{X_k\}_{k=0:P-1}}{\operatorname{argmin}} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} w_{kj} \left\| r_{kj} \right\|^2 \tag{6.2}$$

Here $w_{kj} = 1$ if 3D-point X_k is visible in the j^{th} robot key-frame otherwise $w_{kj} = 0$. Although omitted here for readability, Cauchy loss function has been used in the minimization problem (Equation 6.2) for reducing the effect of noise/outliers on the final estimates. Here K & π are the intrinsic calibration parameters and the fisheye projection model respectively. M robot keyframes (pose + image) are extracted from VO for use in loop closure and bundle adjustment process, and consequently in the next steps of the multi camera registration procedure. A comparison of trajectories before and after bundle adjustment is shown in Figure 6.5.

Because of the asynchronous nature of data generated from several sources, pose interpolation is used to associate a robot pose to each aruco detection. Consider a case when the aruco detection has a timestamp $t \in (t_i, t_j)$ where t_i and t_j are the timestamps of the robot keyframes F_i and F_j with poses $T_{F_i}^W = \begin{bmatrix} R(q_{F_i}^W) & p_{F_i}^W \\ 0 & 1 \end{bmatrix}$ and $T_{F_j}^W = \begin{bmatrix} R(q_{F_j}^W) & p_{F_j}^W \\ 0 & 1 \end{bmatrix}$ respectively, then the robot pose $T_F^W(t)$ at time t is determined by taking weighted average of $p_{F_i}^W$ & $p_{F_j}^W$ - for the translation component (Equation 6.4), and `slerp` (Spherical Linear Interpolation [75]) - for the rotation

component (Equation 6.5).

$$\alpha = \frac{t - t_i}{t_j - t_i} \tag{6.3}$$

$$p_F^W(t) = (1-\alpha)p_{F_i}^W + \alpha p_{F_j}^W \tag{6.4}$$

$$q_F^W(t) = \texttt{slerp}(q_{F_i}^W, q_{F_j}^W, \alpha) \tag{6.5}$$

$$T_F^W(t) = \begin{bmatrix} R(q_F^W(t)) & p_F^W(t) \\ 0 & 1 \end{bmatrix} \tag{6.6}$$

This method associates each aruco detection $T_M^{C_k}(t)$ with an interpolated robot pose $T_F^W(t)$ which helps determine the metric scale s of the robot trajectory from SVO, the spatial offset T_M^F between fisheye camera F and aruco marker M, and to initialize distributed camera poses $\{T_{C_k}^W\}_{k=1:N}$.

6.6.2 Determine the scale of VO using Aruco detections

The aruco detection pose measurements $T_M^{C_k}$ (this is the pose of the marker M in frame C_k) are used to determine the metric scale of the trajectory estimated by monocular VO. The motion-based calibration technique [18] is employed for this purpose, using which the robot motion estimated by VO is aligned with the motion of the aruco marker estimated by its detection in an infrastructure camera C_k. This is similar to the motion based residual developed for LiDAR IMU and Camera IMU registration in Sections 3.4 & 4.4 respectively. For two adjacent robot keyframes F_i & F_{i+1}, the nearest aruco detection measurements $T_{M_i}^{C_k}$ & $T_{M_{i+1}}^{C_k}$ in the same infrastructure camera C_k are searched for by comparing timestamps. Next, the previously described pose interpolation technique is used to determine the robot pose ${}_{int}T_{F_i}^W$ & ${}_{int}T_{F_{i+1}}^W$ corresponding to timestamps of aruco measurements $T_{M_i}^{C_k}$ & $T_{M_{i+1}}^{C_k}$ respectively. The goal here is to align ${}_{int}T_{F_{i+1}}^{F_i}$ $(= ({}_{int}T_{F_i}^W)^{-1}\,{}_{int}T_{F_{i+1}}^W)$ with $T_{M_{i+1}}^{M_i}$ $(= (T_{M_i}^{C_k})^{-1} T_{M_{i+1}}^{C_k})$ and in the process determine the metric scale s of robot trajectory and also the spatial offset T_M^F between the fisheye camera F and the marker M. The VO trajectory is modelled with a scale factor s multiplied to the translation component ${}_{int}p_{F_{i+1}}^{F_i}$. A non-linear least squares optimization problem is solved to determine the unknown variables. The residual r_i

for this optimization process is given by:

$$r_i = {}_{int}T^{F_i}_{F_{i+1}} T^F_M \ominus T^F_M T^{M_i}_{M_{i+1}} \tag{6.7}$$

The symbol $\ominus$ differentiates the operation from a standard subtraction in Euclidean space and is meant to demonstrate operations on manifolds. Here ${}_{int}T^{F_i}_{F_{i+1}} = \begin{bmatrix} R({}_{int}q^{F_i}_{F_{i+1}}) & s.{}_{int}p^{F_i}_{F_{i+1}} \\ 0 & 1 \end{bmatrix}$. Scale s is determined in this step under the assumption that the variable T^F_M is known. This problem (Equation 6.8) is solved by using the ceres solver [5].

$$\hat{s} = \underset{s \in R}{\operatorname{argmin}} \sum_{i=0}^{L-1} \|r_i\|^2 \tag{6.8}$$

6.6.3 Determine the spatial offset between the Aruco marker M and the upward-looking fisheye camera F

The residual r_i in Equation 6.7 is used to determine the spatial offset T^F_M. In this step T^F_M is optimized under the assumption that the variable s in Equation 6.7 is known.

$$\hat{T}^F_M = \underset{T^F_M \in SE(3)}{\operatorname{argmin}} \sum_{i=0}^{L-1} \|r_i\|^2 \tag{6.9}$$

In practice, Equations 6.8 and 6.9 are solved sequentially several times in a loop to arrive at a final estimate of scale s and the spatial offset T^F_M. The VO pose estimate is rescaled using the estimated scale (Figure 6.5), and the estimated spatial offset $\hat{T}^F_M$ is used to initialize the infrastructure camera poses (Section 6.6.4.1). It is important to emphasize here that in this work the calibration robot's trajectory is largely planar, so it is not possible to estimate the complete 6-DoF calibration parameters between the fisheye camera F and the aruco marker M, because motion-based calibration methods require rotation about two non-parallel or non-anti-parallel axes for complete observability [76].

6.6.4 Estimate the poses of the infrastructure cameras

Initialization of infrastructure camera poses is necessary in the non linear optimization based estimation process to ensure convergence to meaningful values. The advantage of the proposed method is that no manual initialization obtained from manual measurements is necessary. The measurements from the infrastructure cameras and visual odometry are enough to perform pose initialization. After the infrastructure camera poses have been initialized, full optimization is undertaken.

6.6.4.1 Initialization

The pose of a camera from the distributed system is initialized using the aruco marker measurements made by it. For a single camera C_k of the distributed system, the pose of the robot corresponding to each aruco measurement made by the camera is inferred by means of the interpolation technique described in Section 6.6.2. For each aruco measurement $T_{M_i}^{C_k}$ made by camera C_k, and the corresponding interpolated robot pose ${}_{int}T_{F_i}^{W}$, the pose of infrastructure camera C_k can be determined from Equation 6.10.

$$T_{C_{k_i}}^{W} = {}_{int}T_{F_i}^{W}\hat{T}_{M}^{F}(T_{M_i}^{C_k})^{-1} \tag{6.10}$$

This operation in Equation 6.10 is performed for N_k measurements from camera C_k and the mean of $\{T_{C_{k_i}}^{W}\}_{i=0:N_k-1}$ is calculated to initialize the pose of camera C_k. For the translation components of $\{T_{C_{k_i}}^{W}\}_{i=0:N_k-1}$ the centroid serves as the mean, and for the rotation components (parametrized as quaternions) quaternion averaging [77] is performed. This operation is repeated for all infrastructure cameras $\{C_k\}_{k=1:N}$ to initialize their respective 6-DoF pose.

6.6.4.2 Optimization

The method described in Section 6.6.4.1 initializes the poses of infrastructure cameras $\{C_k\}_{k=0:N-1}$ by using the aruco detection pose measurements. In this step, the goal is to minimize the reprojection error between the projection $\pi(K, T_{F_j}^{W}, T_{C_k}^{W})$ of the initialized infrastructure camera pose $T_{C_k}^{W}$

and the corresponding pixel detection $\begin{bmatrix} u \\ v \end{bmatrix}_{kj}$ (obtained using OpenCV's blob detection algorithm) on the upward facing fisheye camera image (Figure 6.3a). $\pi(K, T_{F_j}^W, T_{C_k}^W)$ is associated to $\begin{bmatrix} u \\ v \end{bmatrix}_{kj}$ by doing a nearest neighbour search. The residual for minimizing the re-projection error is given in Equation 6.11.

$$r_{kj} = \begin{bmatrix} u \\ v \end{bmatrix}_{kj} - \pi(K, T_{F_j}^W, T_{C_k}^W) \tag{6.11}$$

r_{kj} can be defined as the re-projection error of the k^{th} infrastructure camera when viewed from the j^{th} robot key-frame pose. Here K & π are the intrinsic calibration parameters and the fisheye projection model respectively. This minimization problem (Equation 6.12) is solved using ceres library [5].

$$\{\hat{T}_{C_k}^W\}_{k=0:N-1} = \underset{\{T_{C_k}^W\}_{k=0:N-1} \in SE(3)}{\operatorname{argmin}} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} w_{kj} \left\| r_{kj} \right\|^2 \tag{6.12}$$

Here $w_{kj} = 1$ if infrastructure camera C_k is visible in the j^{th} robot key-frame otherwise $w_{kj} = 0$. The Cauchy loss function is used to suppress the effect of outliers in the minmization problem (Equation 6.12). The optimization is further constrained by minimizing a cost function which uses the epipolar constraint between two infrastructure cameras with a shared FoV as the residual (Equation 6.13).

$$_{\mathbf{I}}\mathbf{r}_{\mathbf{ij}} = {}_{\mathbf{I}}\mathbf{x}_{\mathbf{i}}^{\top} \mathbf{E}_{\mathbf{ijI}} \mathbf{x}_{\mathbf{j}} \tag{6.13}$$

Here, $\mathbf{E}_{ij} = [\mathbf{p}_{C_j}^{C_i}]_\times R_{C_j}^{C_i} = [(R_{C_i}^W)^\top (\mathbf{p}_{C_j}^W - \mathbf{p}_{C_i}^W)]_\times (R_{C_i}^W)^\top R_{C_j}^W$ is the essential matrix between the i^{th} and j^{th} infrastructure cameras with pose $T_{C_i}^W = \begin{bmatrix} R_{C_i}^W & \mathbf{p}_{\mathbf{C_i}}^{\mathbf{W}} \\ 0 & 1 \end{bmatrix}$ & $T_{C_j}^W = \begin{bmatrix} R_{C_j}^W & \mathbf{p}_{\mathbf{C_j}}^{\mathbf{W}} \\ 0 & 1 \end{bmatrix}$ respectively,

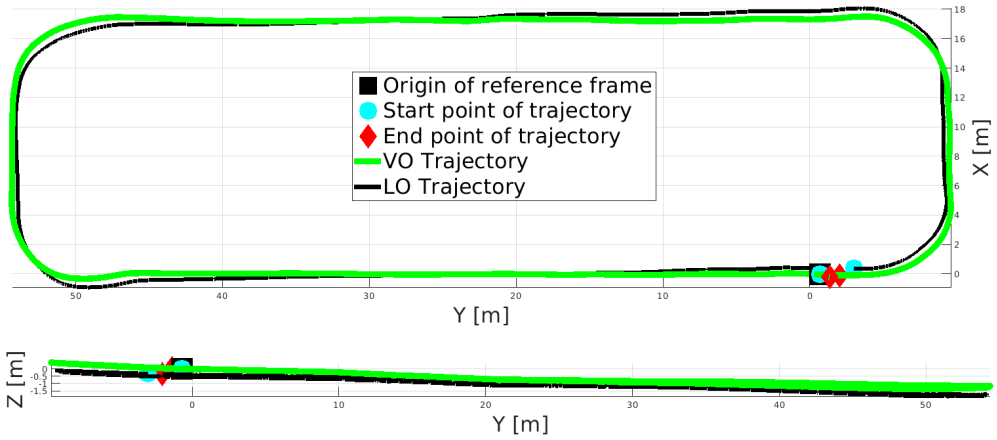
and $\{{}_{\mathbf{l}}\mathbf{x}_{\mathbf{i}}\}_{l=0:3}$ & $\{{}_{\mathbf{l}}\mathbf{x}_{\mathbf{j}}\}_{l=0:3}$ refer to corresponding corners of the aruco marker detected in infrastructure cameras C_i and C_j respectively. The optimization problem from the epipolar constraint (Equation 6.13) is given in Equation 6.14.

$$\{\hat{T}^W_{C_k}\}_{k=0:N-1} = \underset{\{T^W_{C_k}\}_{k=0:N-1} \in SE(3)}{\operatorname{argmin}} \sum_{i \neq j} \sum_{l=0}^{3} w_{ij\,\mathbf{l}}\mathbf{r}_{\mathbf{ij}} \tag{6.14}$$

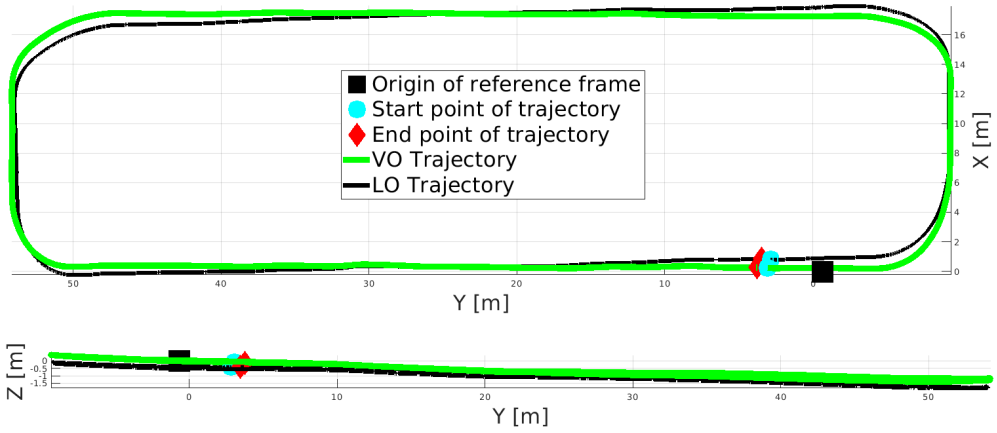
Here $w_{ij} = 1$ when i^{th} & j^{th} infrastructure cameras view the calibration robot simultaneously otherwise $w_{ij} = 0$. The Cauchy loss function is used for robust estimation. In addition to optimizing for infrastructure cameras poses $\{T^W_{C_k}\}_{k=0:N-1}$, the robot keyframe poses $\{T^W_{F_j}\}_{j=0:M-1}$ and the 3D-Map $\{X_k\}_{k=0:P-1}$ are also optimized by solving Equations 6.2, 6.12 and 6.14 together. It is worthwhile to summarize here that the initialization and the determination of final estimates of the infrastructure camera poses not only required the detection of the calibration robot in the infrastructure cameras but also the detection of the infrastructure cameras in the calibration robot (i.e. in its fisheye camera). Thus the variables optimized are constrained by looking both ways. Figure 6.1 shows a plot of estimated variables and aruco detection pose measurements.

6.7 Experiments & Results

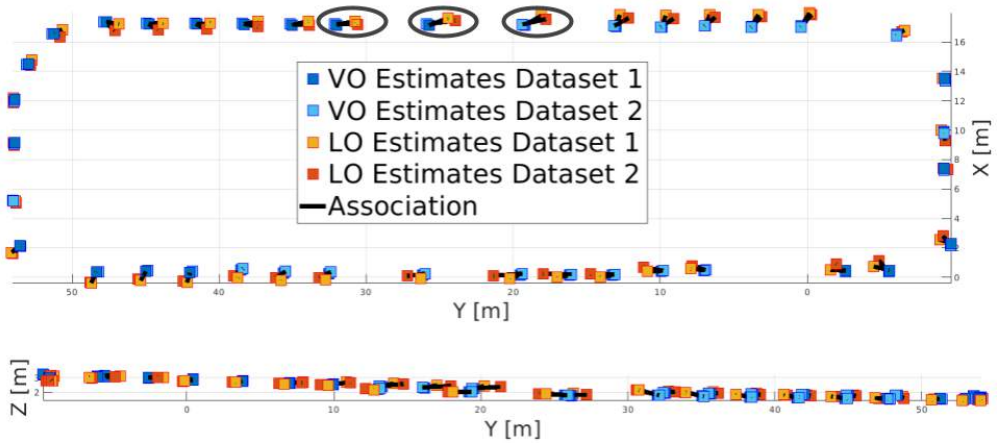
The proposed technique was validated on two datasets by driving the calibration robot under a loop of several infrastructure cameras of the distributed infrastructure camera system. 43 and 38 infrastructure cameras were calibrated in datasets 1 & 2 respectively. As a measure to validate the result of estimation, the robot is driven such that the start and end poses of the trajectory are close to each other. The images from its upward-looking fisheye camera are used to determine robot pose by visual odometry (VO), and the infrastructure cameras detect the aruco marker on the robot (Figure 6.3b). Additionally, 3D scans were also collected from the LiDAR on the calibration robot for comparing the VO based solution against a LO based approach. A plot of the robot trajectories estimated by VO & LO for datasets 1 & 2 is presented in Figures 6.6a & 6.6b. The trajectories are aligned together by aligning the poses of the static infrastructure cameras estimated by both the methods using a technique similar to [78]. The LO based approach determines robot trajectory by



(a) Top and side view of calibration robot trajectories estimated by VO & LO based methods for dataset 1.



(b) Top and side view of calibration robot trajectories estimated by VO & LO based methods for dataset 2.



(c) Top and side view of infrastructure camera positions estimated by VO and LO based approaches for dataset 1 & 2. The ovals drawn around the top longer edge is to highlights the larger difference along $Y-axis$ for the positions estimated from both methods across datasets.

Figure 6.6: **VO vs LO**: Comparison of robot trajectories and estimated infrastructure camera poses between the VO and LO methods. Both methods are represented in a common frame of reference.

Reprojection Error (Pixel)		
	Before Optimization	Post Optimization
Dataset 1	51.642	10.428
Dataset 2	54.245	5.675

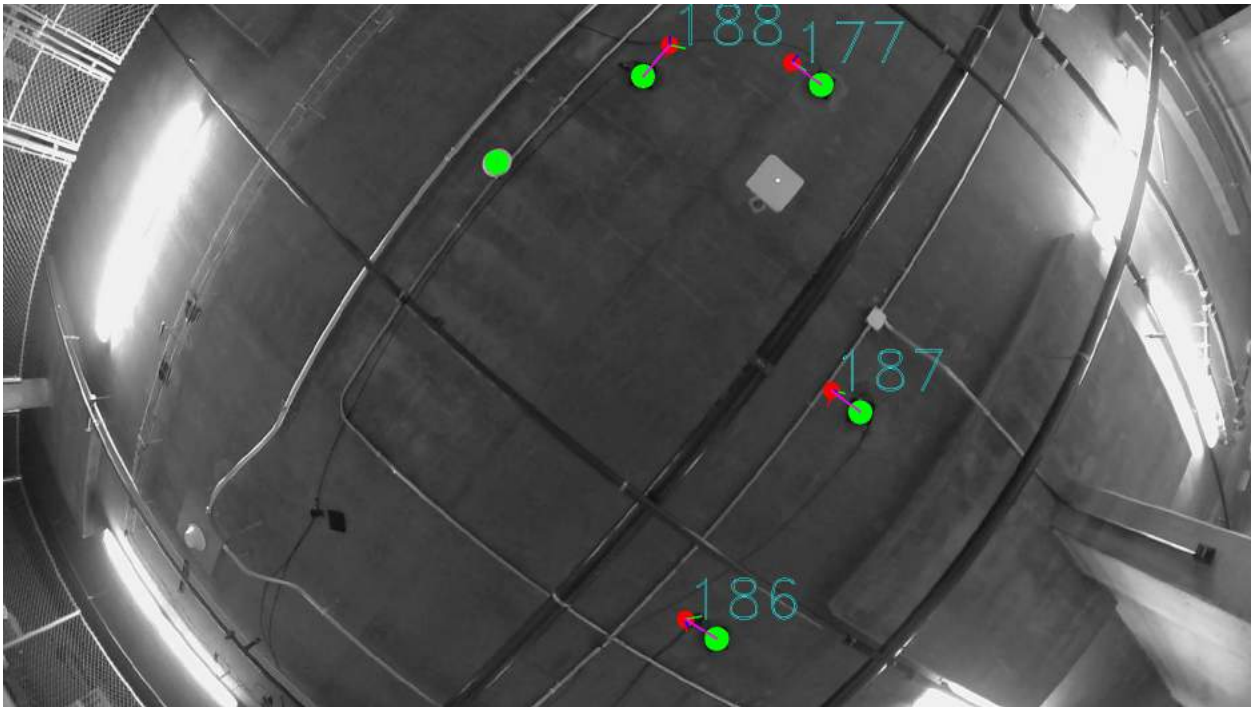
Table 6.1: Root Mean Squared Re-projection Error calculated by measuring the reprojection error of the estimated distributed camera poses on each fisheye camera keyframe.

scan matching [73] and estimates the infrastructure camera poses by minimizing a cost function with a residual similar to Equation 6.10. As LiDAR measures 3D points in metric units, it is not necessary to determine the metric scale of LO, but the spatial offset between the LiDAR and the aruco marker is determined using a method similar to the one described in Section 6.6.3.

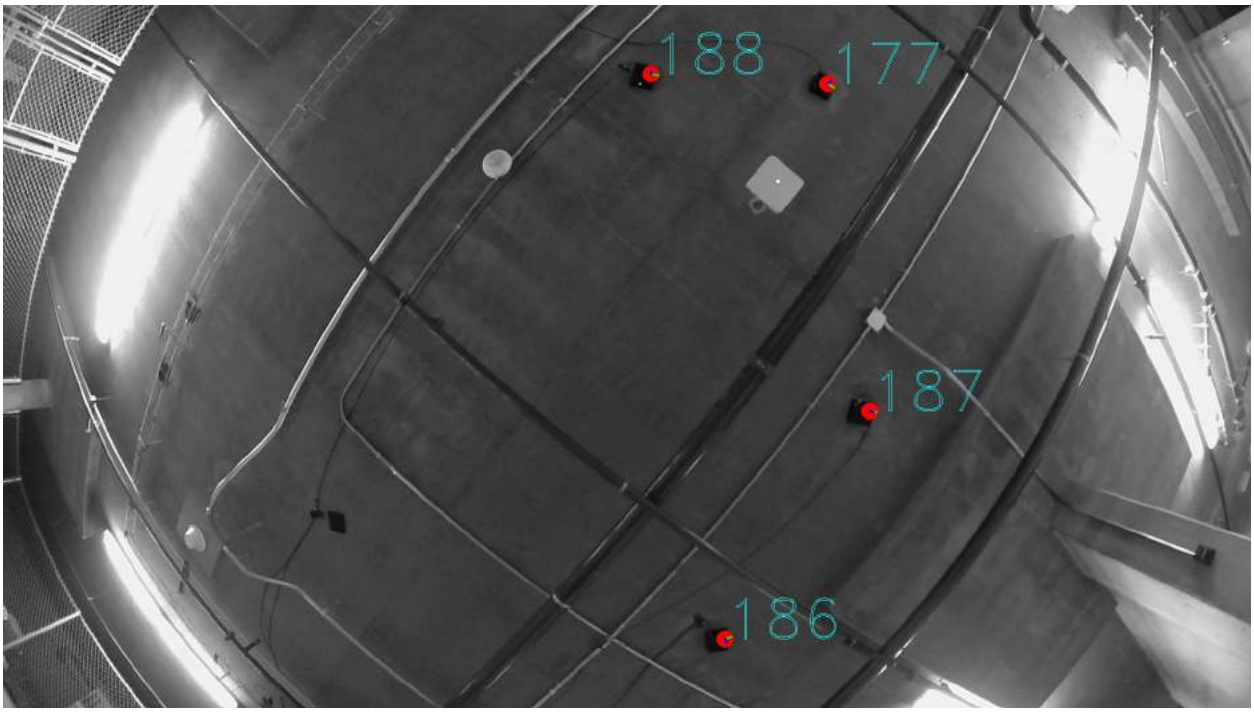
6.7.1 Verification using self-contained methods

To verify the proposed approach, a visual comparison of infrastructure camera re-projection errors before and after optimization is presented in Figures 6.7a & 6.7b. Clearly, the misalignment in the projection of estimated infrastructure cameras and the corresponding detections (using OpenCV's blob detection) in the calibration robot's fisheye camera is significant before optimization (Figure 6.7a), whereas one sees diminished misalignment after the optimization (Figure 6.7b). The Figures 6.7a & 6.7b show the qualitative result for just one robot-keyframe image. The root mean squared re-projection error (RMSE) over the entire robot trajectory for both the datasets is presented in Table 6.1, where it is shown quantitatively that the re-projection error post optimization is significantly smaller.

Next, the extrinsic calibration is validated by detecting the calibration robot in the infrastructure cameras and verifying if the epipolar lines in a pair of infrastructure cameras, with shared FoV & simultaneously looking at the calibration robot's aruco marker, pass through corresponding interest points. The epipolar lines are calculated using the essential matrix between two infrastructure cameras, which is determined by the knowledge of the estimated poses of those two cameras. Therefore, if the epipolar lines pass through the corresponding interest points, it implies that the

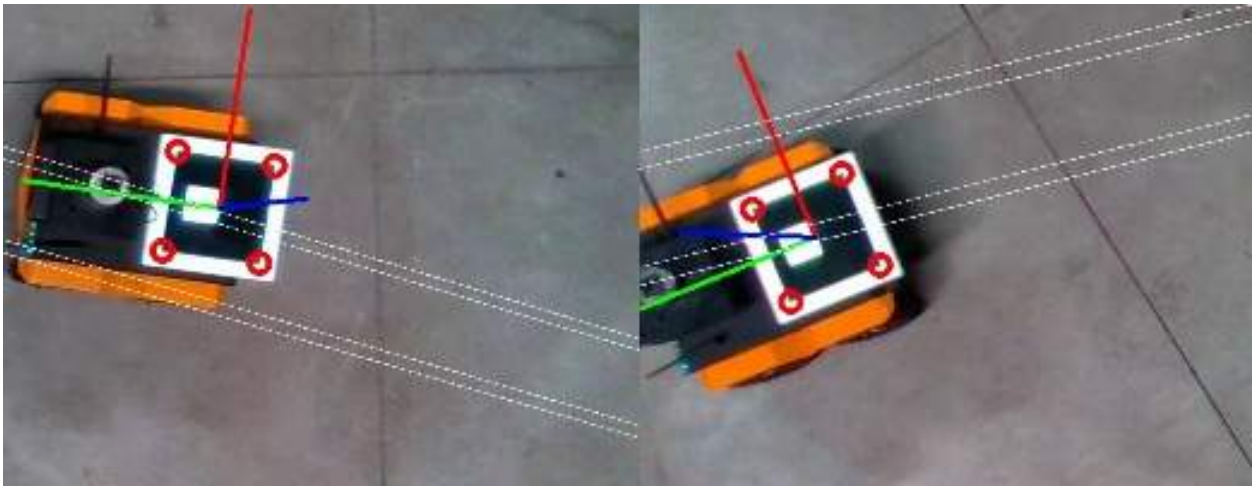


(a) **Misalignment after pose initialization**. The detected infrastructure cameras (green) do not align with projection of initialized infrastructure cameras (red).

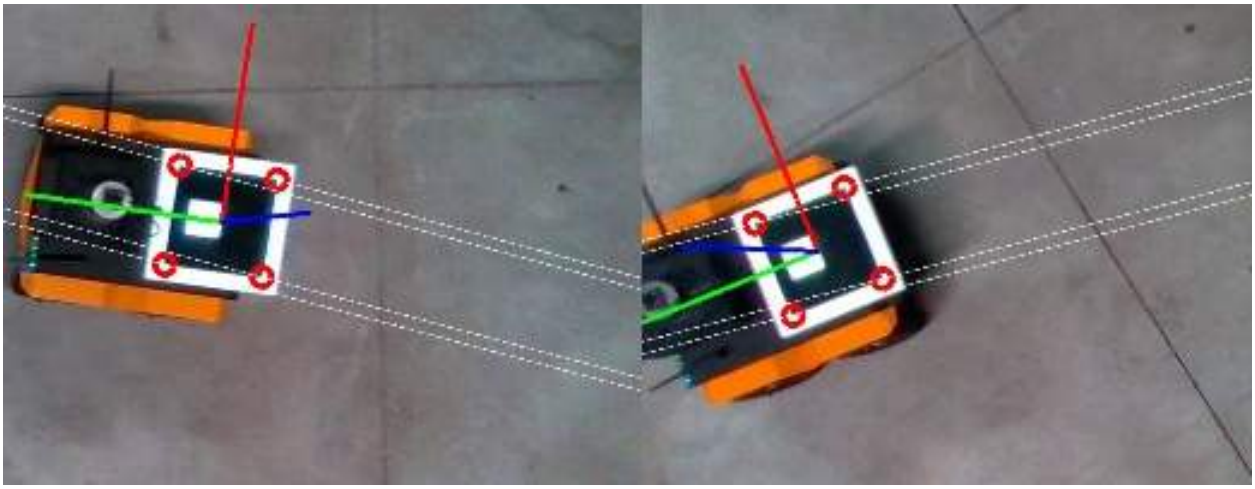


(b) **Re-projection after optimization.** The projection of estimated infrastructure cameras coincide with corresponding infrastructure cameras on the fisheye image.

Figure 6.7: Re-projection of distributed camera poses before and after solving the global optimization problem (Section 6.7).



(a) **Epipolar lines before optimization.**



(b) **Epipolar lines after optimization.**

Figure 6.8: Epipolar lines drawn, on a pair of images from infrastructure cameras with shared FoV, before and after solving the global optimization problem.

relative pose between the two cameras is correct (to scale). The interest points in the current context happen to be the four corners of the detected aruco marker on the calibration robot.

In a well-textured environment, it would have been possible to detect and match feature points between two views to verify pair-wise extrinsic calibration but the scene in the present scenario is largely texture less as shown in Figures 6.8a & 6.8b, so the corners of the aruco marker are used instead as features to track across frames. As shown in Figure 6.8a, the epipolar lines calculated from unoptimized infrastructure cameras poses do not pass through the corresponding interest points,

Sampson Distance		
	Before Optimization	Post Optimization
Dataset 1	0.156144	2.11468e-05
Dataset 2	0.17483	2.31116e-05

Table 6.2: RMS Sampson distance calculated by using simultaneous viewing of the calibration robot from pair of infrastructure cameras over several pairs.

whereas after optimization these lines pass through the corresponding interest points as shown in Figure 6.8b. While Figures 6.8a and 6.8b provide visual verification of extrinsic calibration for one pair of infrastructure cameras which share FoV, quantitative results are presented in Table 6.2 where the mean sampson distance over several infrastructure cameras pairs for both the datasets has been tabulated. Sampson distance is a first-order approximation to the geometric error between an interest point and the corresponding epipolar line. In Table 6.2 it can be seen that the average Sampson distance before the optimization is larger than the Sampson distance post-optimization in both datasets 1&2.

6.7.2 Comparison with Lidar Odometry based method

In the absence of ground truth, the proposed method is compared against a LO based approach with the goal to verify if the VO based solution presented here gives comparable results. In order to compare the absolute infrastructure camera poses the estimated variables are represented in a common frame of reference. The reference frame of the VO based solution from dataset 1 - W_1^{VO} is chosen as the reference for comparison. Next, the transformations $T_{\chi}^{W_1^{VO}}$ to transform the infrastructure cameras poses estimated in frames $\chi = \{W_2^{VO}, W_1^{LO}, W_2^{LO}\}$ to W_1^{VO} are determined by minimizing the cost function in Equation 6.15 which is a method similar to [78] using ceres-solver [5].

$$\hat{T}^{W_1^{VO}} = \underset{T_{\chi}^{W_1^{VO}}}{\operatorname{argmin}} \sum_{k=0}^{N-1} \left\| T_{C_k}^{W_1^{VO}} \ominus T_{\chi}^{W_1^{VO}} T_{C_k}^{\chi} \right\|^2 \tag{6.15}$$

RMSE between corresponding infrastructure cameras						
	ϕ°	θ°	ψ°	X[m]	Y[m]	Z[m]
$LO_1 \leftrightarrow VO_1$	3.7518	3.4510	3.0424	0.4509	0.6568	0.1388
$LO_1 \leftrightarrow VO_2$	3.2555	3.7371	2.7007	0.4625	0.7684	0.1375
$LO_2 \leftrightarrow VO_1$	3.5204	3.6040	3.6998	0.4095	0.8709	0.1328
$LO_2 \leftrightarrow VO_2$	3.1391	3.7304	2.0995	0.4371	0.9697	0.1330

Table 6.3: RMSE for estimated env. camera poses between VO and LO based approaches for both the datasets. Here LO_i : infrastructure cameras estimated from LO based approach in dataset i & VO_j : Corresponding infrastructure cameras estimated from VO based approach in dataset j.

The infrastructure camera C_ks' poses are used for determining $T_\chi^{W_1^{VO}}$ instead of aligning VO & LO trajectories because the cameras are static landmarks with unique ids that solve the data association problem across datasets and methods (VO/LO). Moreover, aligning trajectories involves matching timestamps which may not be accurate as the camera and LiDAR are neither triggered simultaneously nor synced with each other. Using the estimated $T_\chi^{W_1^{VO}}$, the infrastructure cameras are transformed to a common frame of reference for comparing absolute infrastructure camera poses. Visualization of the infrastructure camera positions for both the approaches and both datasets is presented in Figure 6.6c. The RMS difference between corresponding infrastructure cameras poses estimated using both the VO based approach and the LO based approach for both the datasets are presented in Table 6.3, where it's observed that the RMS translation difference varies between 0.41-0.46 m along X axis and 0.1330-0.1388 m along Z axis, the difference is larger and varies between 0.66-0.97 m (a greater range) along Y axis. It's observed in Figure 6.6c that the difference between the estimated infrastructure camera poses between VO & LO across both the datasets is more along the longer edges of the elliptical trajectory. More specifically, the difference is larger at the top long edge with maximum misalignment being along Y axis as validated in Table 6.3 and highlighted ovals in Figure 6.6c. The maximum translational difference is $\sim$ 1.08 m over a loop of 150-155 m and the maximum angular difference is 3.75°. As the difference is not systematic, one can attribute this to local measurement/estimation errors or local scale divergence in one or both

Sampson Distance		
	LO	VO
Dataset 1	0.0950822	2.11468e-05
Dataset 2	0.0849663	2.31116e-05

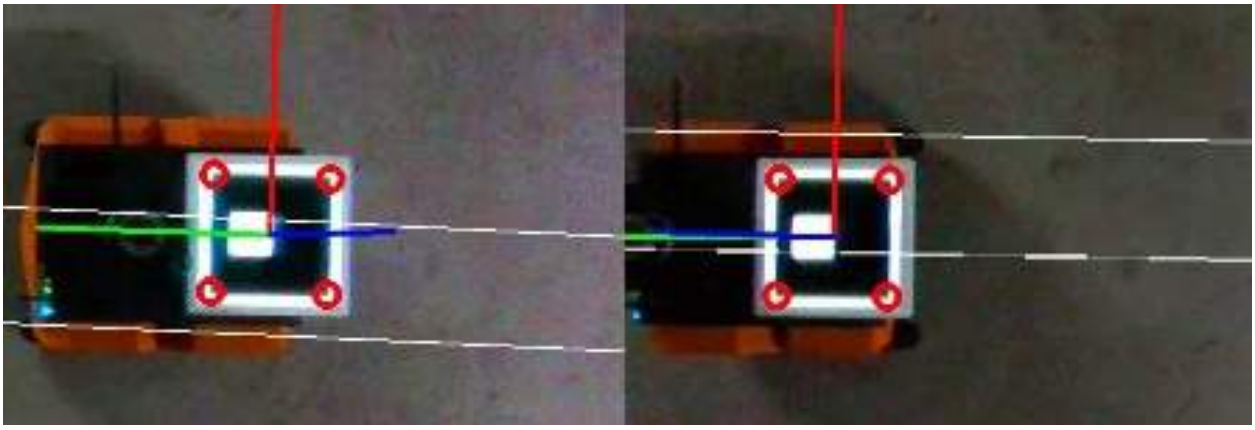
Table 6.4: Tabulation of RMS Sampson distance for pairs of infrastructure cameras which simultaneously viewed the calibration robot's aruco marker. It's see that the Sampson distance is lower for VO based approach across both the datasets.

the estimation techniques.

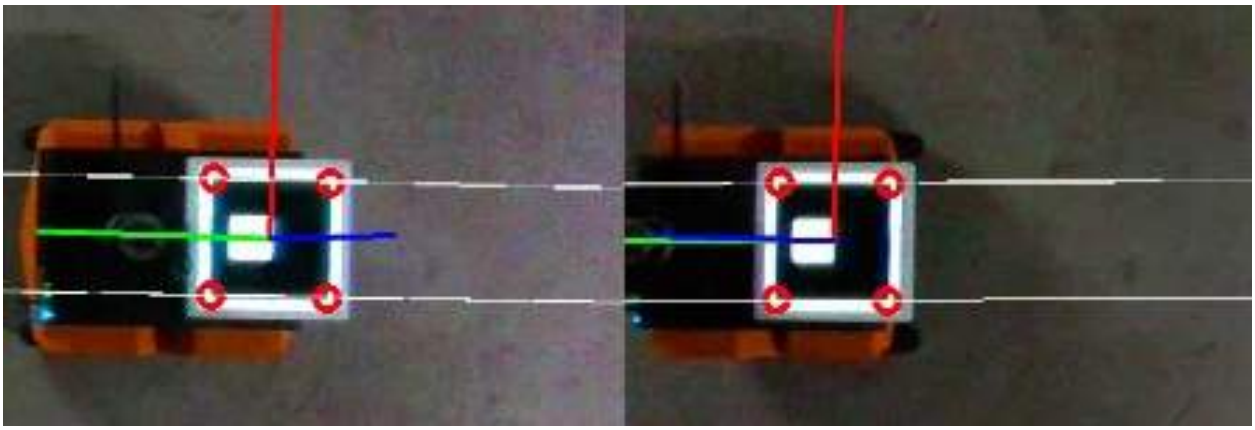
Next, the extrinsic calibration between pairs of infrastructure cameras that simultaneously view the calibration robot's aruco marker has been compared using the infrastructure camera pairs' pose estimates which determine the essential matrix between the viewing pairs. The corresponding aruco marker's corners are detected in both the viewing cameras and the corresponding epipolar lines are drawn by using the aforementioned essential matrix and the detected corners. The Sampson distance between the detected points and the corresponding epipolar line gives us a measure of the quality of the extrinsic calibration. In this comparison, it is seen that the VO based approach fares better as evident from comparing Figure 6.9a against Figure 6.9b. The RMS Sampson distance from both the methods for both the datasets has been tabulated in Table 6.4 - from which it may be conclude that in the VO based approach the Sampson distance calculated using estimated infrastructure cameras poses is lower.

6.7.3 Repeatability of Results

In Figure 6.6c it can be observed that the infrastructure camera estimates from the VO approach are bunched closer (light and deep blue squares) to each other than the estimates from the LO approach (light and deep orange squares). This is also quantified in Table 6.5 where the VO based estimates from both the datasets have lower RMS differences when compared against LO based estimates, implying that the VO based approach gives more repeatable results than the LO based method.



(a) **LO solution:** Epipolar lines drawn using a pair of neighboring infrastructure cameras pose estimates do not pass through the corresponding corners.



(b) **VO solution:** Epipolar lines pass through corresponding corners. There is a small overlap in the FoV of these cameras.

Figure 6.9: Epipolar lines drawn using infrastructure cameras pose estimates for both LO and VO based methods for dataset 2. Similar results were seen for dataset 1 which have been omitted here in the interest of space.

RMSE between corresponding infrastructure cameras poses						
	ϕ°	θ°	ψ°	X[m]	Y[m]	Z[m]
$VO_1 \leftrightarrow VO_2$	0.0202	0.1010	0.0870	0.0838	0.1164	0.0379
$LO_1 \leftrightarrow LO_2$	0.0162	0.0069	0.0155	0.2749	0.4013	0.0446

Table 6.5: RMS difference between corresponding infrastructure cameras estimated from two different datasets using the same approach. It is observed that the VO based method gives more repeatable results as evident from the lower RMSE.

6.8 Discussion

The method presented here can be used to perform extrinsic calibration of a large network of infrastructure cameras using a calibration robot equipped with an upward-facing fisheye camera and a backlit marker. In this work a loop of $\sim$40 cameras is calibrated in the perimeter of $\sim$150-155 m and experimental results have been provided on two different datasets with comparison against a LiDAR-based approach (LO). In contrast to LO, two-way sensing is possible with the visual approach, i.e. not only do the infrastructure cameras detect the robot but the infrastructure cameras are also detected in the robot's fisheye camera. This two-way sensing constrains the optimization better. The infrastructure cameras also present unambiguous landmarks with no problems with data association between successive runs and can be used for downstream tasks like robot localization and navigation.

The VO solution is not only comparable to LO, but in some experiments better (Figures 6.9a $\&$ Figures 6.9b, Table 6.4) and more repeatable (Table 6.5). Moreover, the VO method also allows to self-verify the extrinsic calibration by projecting the estimated infrastructure camera position onto the fisheye camera image (Figures 6.7a $\&$ Figures 6.7b). This is not possible in the LO approach. There exist several additional practical issues with LO. First, scan matching shows degradation in performance $\&$ scale divergence in our long corridor-like environment. Second, LiDAR being a spinning sensor outputs point clouds with motion distortion especially while turning. Such distortion can be corrected with additional sensors like IMUs but the goal is to reduce the number of sensors used. Third, it was also found experimentally that the LO scan matching and place recognition (for loop closure) is greatly affected in the indoor testing facility with several dynamic objects around at all times. In contrast, an upward-facing camera mainly tracks static features on the ceiling unaffected by dynamic obstacles and more durable for long-term VO. Most importantly, a camera is smaller, lighter, cheaper, and consumes less power, making its use possible with cheaper robots.

6.9 Conclusion

In conclusion, this work presents (the author believes) the first large-scale calibration of a camera network with a mobile robot equipped with a single camera. The proposed batch optimization framework not only estimates the poses or the static registration parameters of the infrastructure cameras but also the evolving pose of the calibration robot sensor along with a map of 3D features tracked using frame to frame visual odometry. The framework also used place recognition to close loops and performs global bundle adjustment to reduce drift in calibration robot pose. The reduction in drift not only improves robot pose but also the poses of the static infrastructure cameras. Although the purpose of the application presented in this chapter was to register infrastructure cameras, the framework can be easily extended to estimate poses of multiple static or moving sensors that can detect each other in their measurements. The calibration robot detects the infrastructure cameras using OpenCV blob detection and the infrastructure cameras detect the calibration robot via detection of the Aruco marker on the calibration robot. This two way detection plus tracking of visual features in the environment forms a registration problem where the poses of multiple sensors are determined by minimizing several cost function together. Although the estimation framework in the current chapter is different from the filtering technique presented in Chapters 3, 4, 5, the physical quantities estimated are similar. In both the frameworks the variable of interest that are estimated are the extrinsic calibration or the static registration parameters, but in addition to these other evolving proxy variables like pose and velocity of sensors have also been estimated as a part of the estimation algorithm. These proxy variables cannot be eliminated and need to estimated along with the extrinsic calibration parameters. So, both in the filtering frameworks of Chapter 3, 4, 5 and the smoothing framework of the current chapter the estimation algorithm is primarily a multi-sensor calibration, localization and mapping algorithm that estimates poses of all the sensors that generate measurements and contribute to the estimation process. The quality of the poses estimated depend on the data collected for the process. If the user wants to prioritize extrinsic calibration then they must ensure that the data to be fed to the estimation pipeline is such that the extrinsic calibration will be observable. Observability studies are often specific to the system under

consideration and will be an area of future research for the case studied in this chapter.

As far as this application is concerned, future work will explore the extrinsic calibration of a similar system with a camera-equipped drone instead of a camera-equipped ground robot, and the use of such a system for the autonomous navigation of autonomous vehicles (AVs), wheeled and flying robots in controlled settings like warehouses, factory floor, parking lots, etc.

The current chapter presented a method to register several static cameras with respect to a fixed frame by bridging their pose using a calibration robot. The next chapter presents a method to register a single fisheye camera in a prior map for augmenting situational awareness of autonomous vehicles in complex traffic intersections.

7. REGISTRATION OF FISHEYE CAMERA IN A PRIOR MAP FOR AUTONOMOUS VEHICLES

7.1 Introduction

Thus far, the thesis discussed techniques of registering sensors which either involve alignment of common features in a pair of sensors or alignment of motion segments between same time instants obtained from individual pose estimation algorithms in a pair of sensors. This chapter presents registration of a fisheye camera in a prior map comprising satellite imagery and LiDAR mapping data by alignment of probability distribution of sensor measurements. In the real world certain physical objects are measured by different techniques resulting in observations that do not necessarily map to the same values, but their underlying probability densities are essentially the same. This can be demonstrated by looking at RGB image from satellite data and LiDAR reflectance values in Figures 7.3a & 7.3b respectively, where one can visually guess that the number of dark pixels in the satellite image will be roughly equal to the number of dark pixels in the LiDAR internsity image, and the number of bright pixels in the satellite image will be roughly equal to the number of bright pixels in the LiDAR intensity image. The values of dark and bright pixels may map to different ranges in both the images but since the satellite image and the LiDAR intensity image precisely map the same area, the normalized probability distributions of both the images will be the same. This idea can be used for registering two different sensors measuring the same physical quantity in their respective sensing modality. The alignment of probability distribution is obtained by maximization of mutual information between data obtained from two different sensors. In this Chapter, the normalized probability distributions of fisheye image gray scale values and LiDAR intensity values are used to register the fisheye image in the prior LiDAR mapping data. Registration using maximization of mutual information requires good initial values of the registration estimates which is obtained in this chapter by performing feature matching between rectified fisheye image and satellite mapping data followed by solving a PnP problem

(Figure 7.6). Once the registration estimates are initialized a grid search is performed around it and the value of the registration parameter which yields the maximum mutual information is chosen as the correct value of registration/localization of the fisheye image (Figure 7.2a) in the prior map (Figure 7.3). As already presented in various registration algorithms in the previous chapters, data from two different sensors are used to optimize a criteria (mutual information in the current context), the result of which is the 6 DoF pose between the sensors being registered. The difference between the previous and the current chapters is that the previous approaches optimize cost functions that align sparse geometrical features extracted from dense sensor data, while the current chapter aligns probability distributions derived from all sensor data.

7.2 Motivation & Overview

Environment perception in Autonomous Vehicles (AV) is a challenging problem. With the prevalent approach of using only on-board sensors to solve the perception problem, it is difficult to sense occluded areas and mitigate the effects of sensor outage. Complex traffic intersections with buildings close to the curb may minimize the field of view of an AV's sensors. Integration of smart infrastructure nodes (sensing and compute) on roads where AVs operate can help overcome these challenges. The elevated and static view-point of the smart sensors enables them to observe the environment, detect more objects in the scene, and communicate that information to AVs. AVs can fuse the aforementioned information with their own sensor measurements and augment their situational awareness. fisheye cameras are well known for their low cost and wide Field of View (FoV), making them suitable for such smart infrastructure based sensing applications. However, the fisheye camera needs to measure this information in the same coordinate frame as the vehicle. For this reason, they need to be localized or registered within the same map that is used by the AV for navigation. This work proposes a method to localize a downward looking static smart infrastructure fisheye camera in a prior map consisting of a metric satellite image, and a co-registered LiDAR map of ground points with their LiDAR reflectivity values. An overview of the approach is shown in Figures 7.1 & 7.4.

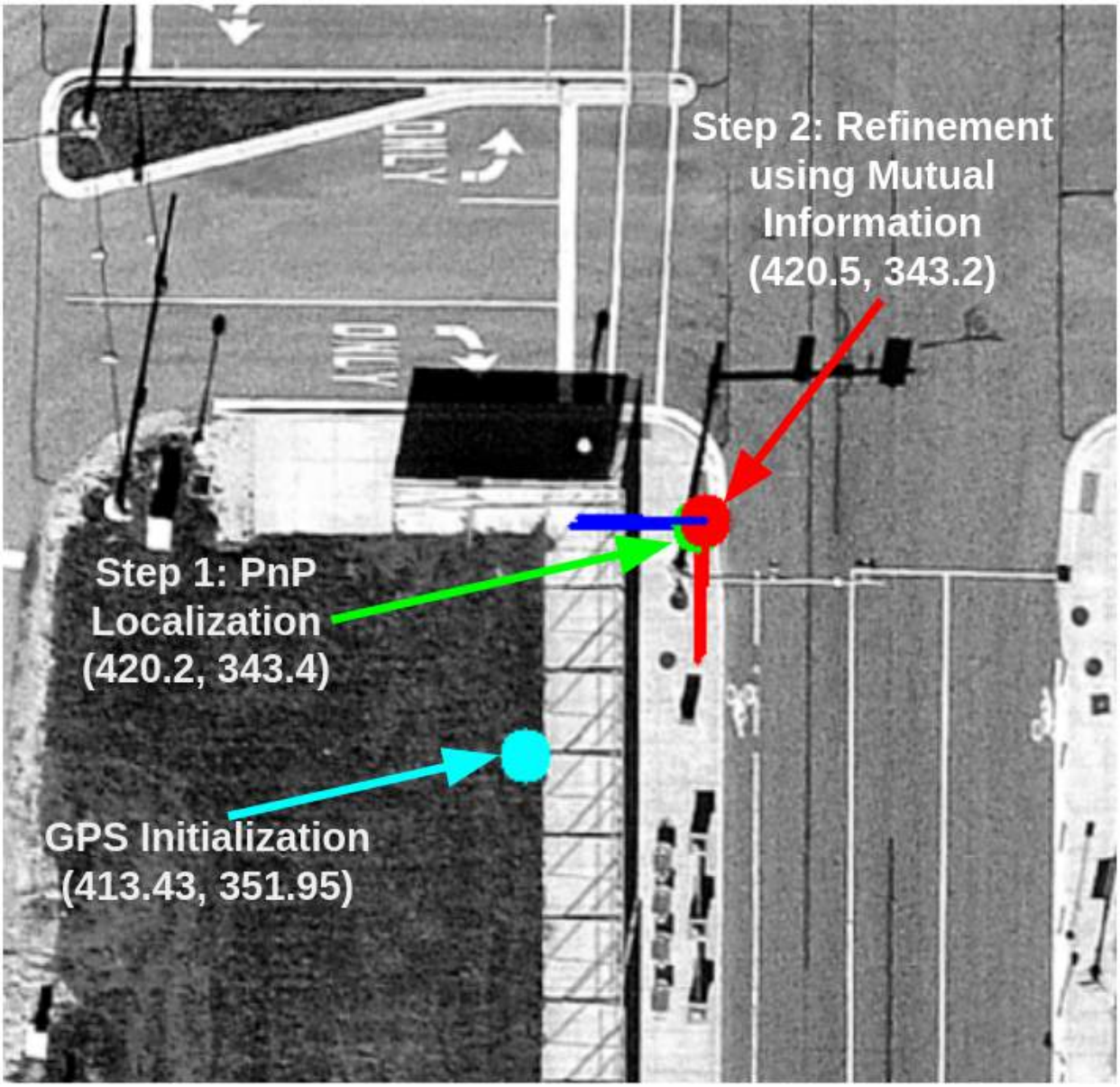


Figure 7.1: Overview of the two step approach to camera registration in prior map. The coordinates shown are in meter, wrt. the origin of the map. The process starts with a noisy GPS initialization (cyan), then feature matching between satellite image and rectified fisheye image is performed to obtain an initial camera pose - PnP Localization (Green), and finally the Mutual Information between LiDAR map and the fisheye image is maximized to obtain a refined camera localization estimate.

7.3 Literature Review

Registration of one sensor in another sensor data is a problem similar to extrinsic calibration of two sensors. The topic of registration of sensors in prior map is gaining importance because of readily available pre-mapped data of cities and urban spaces with the ultimate goal to do only localization and minimal mapping while driving. In the context of extrinsic calibration, the method is similar to target less appearance based extrinsic calibration of multiple sensors. As smart infrastructure based distributed sensing is gaining popularity, the method of registering sensors in a prior map will find a number of applications because one will be able to easily register or localize the infrastructure sensors in the prior map using mapping data and the infrastructure sensor data only without depending on external or auxiliary absolute positioning information which may not be available under certain circumstances (like tunnels, deep urban canyons, etc).

There are several contributions on localization of camera images in prior maps (satellite imagery or LiDAR generated 3D maps). Satellite maps can be procured easily from third party sources [79, 80], and with the widespread use of LiDARs in autonomous driving, there now exist several high definition map providers which provide dense 3D map of the environment [81]. [82] presents LiDAR map based monocular camera localization in urban environment. Unlike [83] which depends on detection of geometric primitives like lines in both the sensing modalities, [82] uses dense appearance based approach which work in unstructured environments. In [82], given an initial belief of the camera pose, they generate several synthetic views of the environment by projecting the LiDAR map points using a perspective camera model, and compare these synthetic views against the live camera feed. The synthetic view which maximizes the Normalized Mutual Information (NMI) between the real image gray scale values and the projected points' LiDAR reflectivity values, is the solution to the localization problem. [82] draws inspiration from work done on 3D-LiDAR Camera extrinsic calibration described in [27], which uses maximization of Mutual Information (MI) for calibrating a 3D-LiDAR Camera pair. [84] presents a camera localization technique which matches ground imagery obtained by cameras onboard an AV to the available satellite imagery. The camera images are warped to obtain a bird's eye view (BEV) of the ground.

Next, the BEV image is matched with the given satellite imagery using using SIFT [85] features.

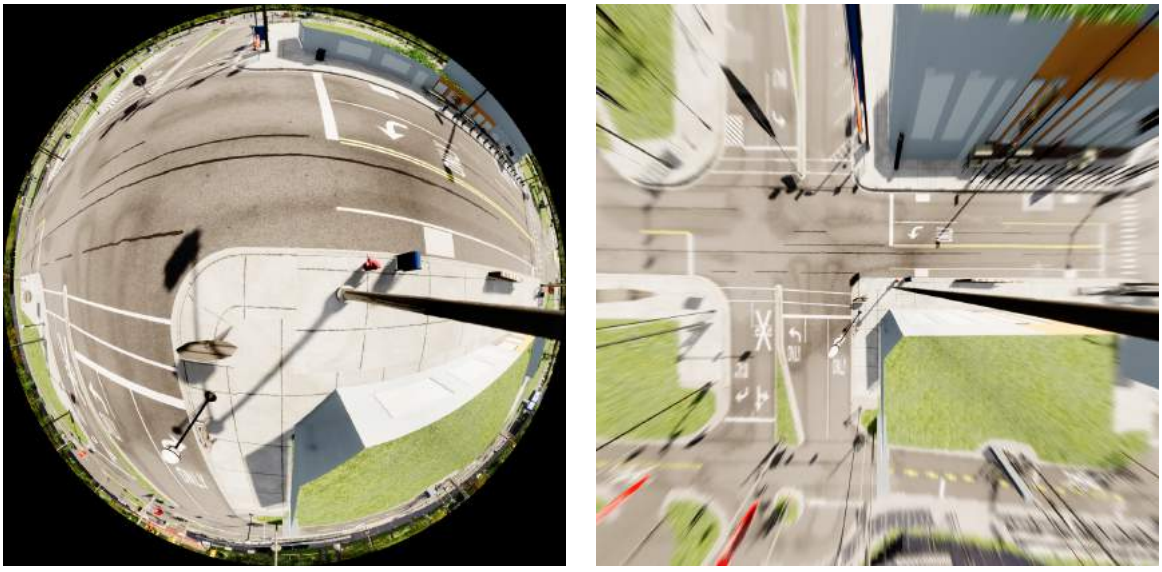
7.4 Contributions

The approaches to register cameras in prior maps described in Section 7.3 provide solutions for perspective cameras, the focus in this Chapter is to register (estimate $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$ in Equation 7.1) a downward looking static fisheye camera (Figure 7.2a) in a prior map (Figure 7.3) which consists of 2D satellite imagery with metric information (Figure 7.3a) and 3D LiDAR map of ground points (Figures 7.3b, 7.3c). The camera registration is initialized using feature matching as done in [84], and refined using maximization of a MI based cost function as done in [82] and [27].

7.5 Overview

This section provides an overview of the various components of the system.

7.5.1 fisheye Camera



(a) fisheye image captured at a traffic intersection in our simulator

(b) Rectification of fisheye image to perspective image

Figure 7.2: fisheye Image (Figure 7.2a) and its perspective rectification (Figure 7.2b)

The fisheye camera projection model proposed in [12] is used to project 3D point $\mathbf{P_W} = [X_W, Y_W, Z_W]$ defined in the prior map coordinate frame $\mathbf{W}$ to a 2D point $\mathbf{p}$ on the fisheye camera image plane using Equation 7.1.

$$\mathbf{p} = \mathbf{\Pi}(K, D, \xi, [{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}], \mathbf{P_W}) \tag{7.1}$$

Here $\mathbf{\Pi}()$ is the projection function, $K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \end{bmatrix}$, $D = [k_1, k_2, p_1, p_2]$ and ξ are the camera intrinsics, and $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$ is the camera extrinsic. ${}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}} \in SO(3)$ is an orthonormal rotation matrix and ${}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}} \in R^3$ is a 3D vector. The goal of this work is to estimate the unknown camera extrinsic $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$ in the prior map frame $\mathbf{W}$.

7.5.1.1 Intrinsic Calibration

The intrinsic parameters K, D and ξ are estimated by collecting several images of a large checkerboard at different poses, and feeding those images to the omnidirectional camera calibrator in OpenCV[49], which provides an implementation[1] of the intrinsic calibration technique presented in [86].

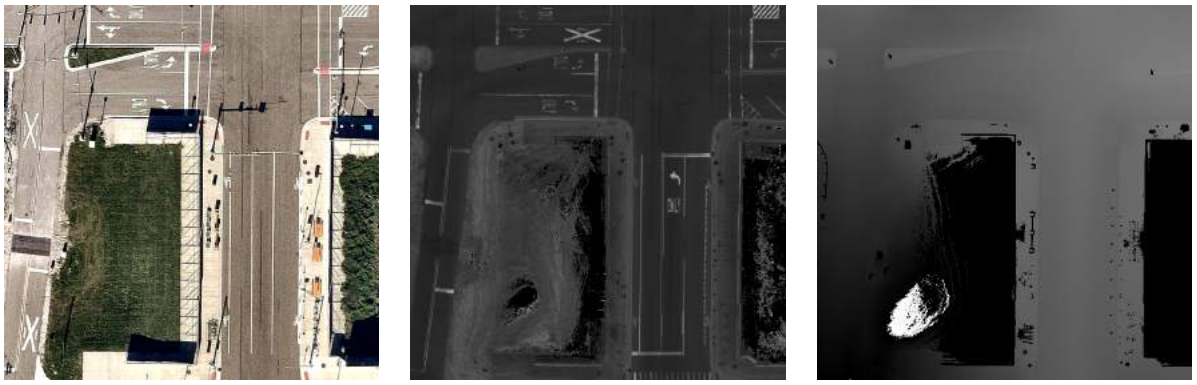
7.5.1.2 Rectification

The intrinsic camera calibration parameters are used to rectify the fisheye image (Figure 7.2a) into its corresponding perspective image (Figure 7.2b) utilizing OpenCV's rectification routines. Although perspective rectification results in loss of field of view, it makes the application of common third party computer vision algorithms developed for perspective images possible for fisheye images.

7.5.2 Prior Map

The prior map (Figure 7.3) consists of two important components which are registered and expressed in a common frame of reference $\mathbf{W}$. They are:

[1] `https://docs.opencv.org/4.5.2/dd/d12/tutorial_omnidir_calib_main.html`



(a) Satellite Map

(b) LiDAR Map: Reflectivity of Ground Points

(c) LiDAR Map: Height of Ground Points

Figure 7.3: **Prior Map:** Figures 7.3a, 7.3b & 7.3c show the components of the prior map. The full map is not shown in the interest of space.

7.5.2.1 Satellite Map

The satellite map is a metric Bird's Eye View (BEV) satellite image (Figure 7.3a). For the data used in this thesis, a pixel on the satellite image corresponds to 0.1 m on the ground.

7.5.2.2 LiDAR Map

The prior LiDAR map is built using an offline mapping process described in [82]. Broadly, a survey vehicle equipped with several 3D LiDAR scanners and a high end Inertial Navigation System (INS) is manually driven and sensor data is collected in the environment that needs to be mapped. Next, an offline pose-graph optimization SLAM (Simultaneous Localization and Mapping) problem is solved to obtain the accurate global pose of the vehicle. Finally, a dense ground point mesh is constructed from the optimized pose graph using region growing techniques which gives a dense 3D point cloud map. The ground points from this dense 3D cloud are used to generate the LiDAR ground reflectivity image (Figure 7.3b) and ground height image (Figure 7.3c). The LiDAR Map (Figures 7.3b and 7.3c) is aligned with the satellite imagery (Figure 7.3a) using the GPS measurements from the INS.

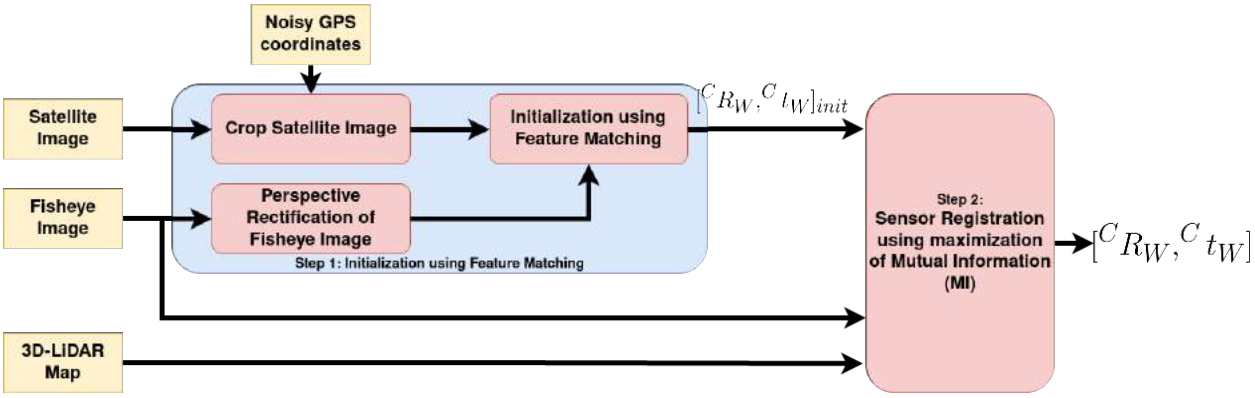


Figure 7.4: **Overview of the method:** The block diagram shows the two steps involved in the registration process. In Step 1, features are matched between the perspective projection of fisheye image and a cropped satellite map to initialize camera pose, and in Step 2 the initialization is refined by maximization of Mutual Information between fisheye image and prior 3D-LiDAR map.

7.6 Problem Formulation

The goal of this work is to estimate the unknown camera pose $[^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, ^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$ in the prior map frame $\mathbf{W}$. It is assumed that a noisy estimate of the fisheye camera's (GPS) position (no orientation) is available which helps to reduce the search space in the prior map. A two step approach is followed to register the camera in the prior map, the details of which are presented in Sections 7.6.1 and 7.6.2, and a broad overview is provided in Figure 7.4.

7.6.1 Initialization using sparse feature matching

Traditional geometric computer vision libraries offer a plethora of feature detectors and descriptors like SIFT[85], SURF[87], ORB[88], etc. which have been used in Visual Odometry/SLAM [67] for estimating camera pose. Lately, there has been a preponderance of learnt features and feature matching techniques, eg. SuperPoint [89] and SuperGlue [6] respectively, that have proven to be robust to viewpoint changes when compared against traditional approaches. Traditionally available feature detection, description and matching techniques are usually suitable for perspective images only. Therefore, the fisheye image is rectified into the corresponding perspective image as explained in Section 7.5.1.2, and SuperGlue [6], a pre-trained deep learning based feature matching algorithm, is used for matching features (Figure 7.5) between the rectified

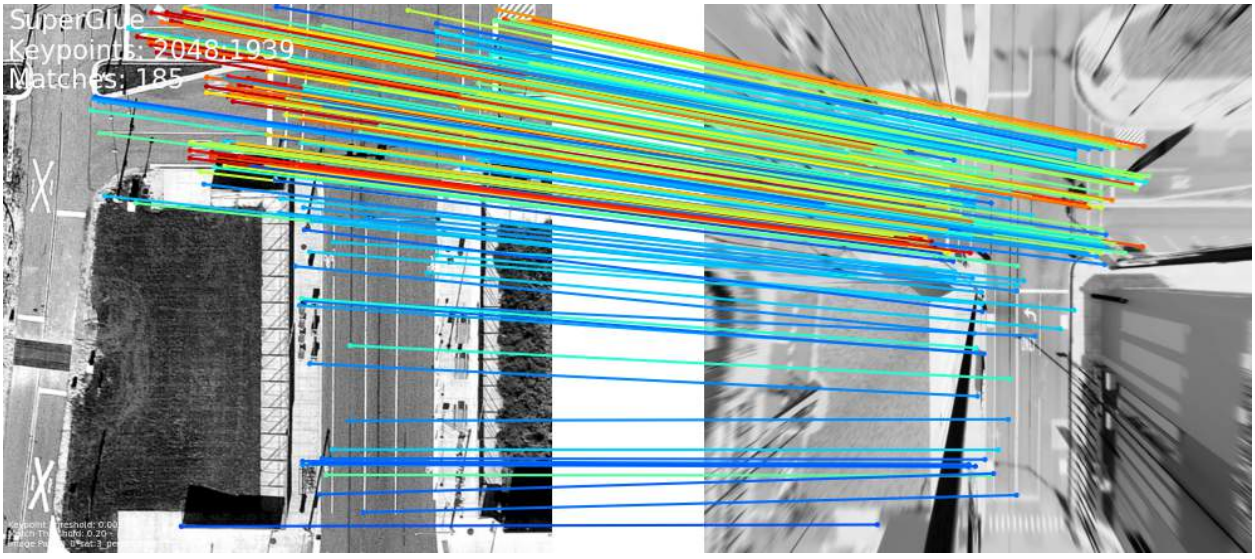


Figure 7.5: SuperGlue [6] Matching between Satellite Image (Left) & perspective projection of fisheye image (Right)

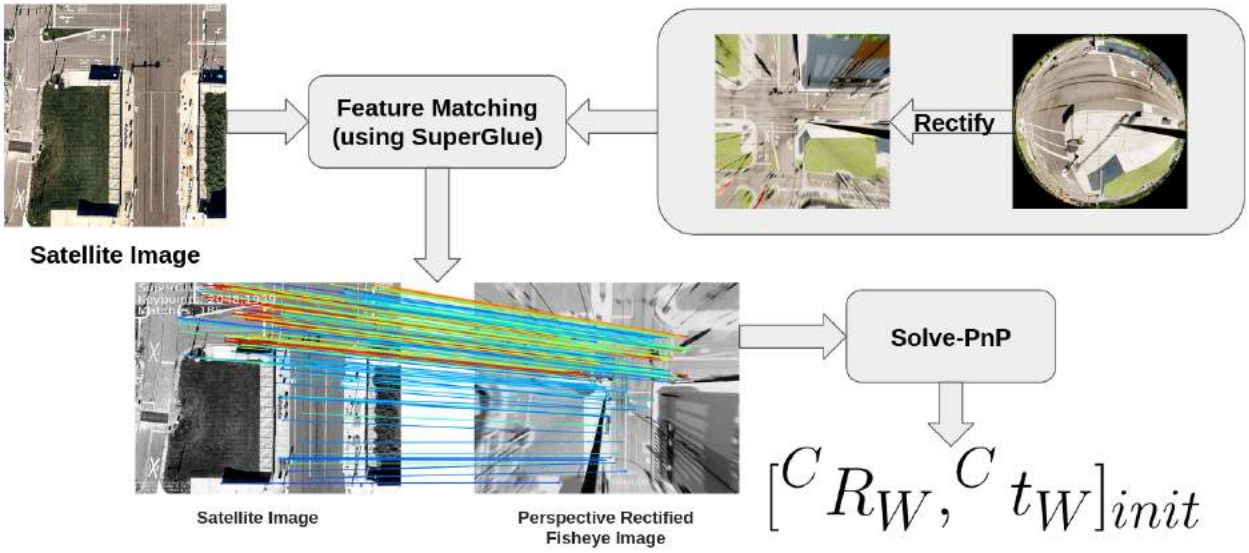


Figure 7.6: The fisheye Camera Image is rectified first, and then SuperGlue based deep feature matching is performed between the metric satellite image and the rectified fisheye image. The matched features are used to solve a PnP problem using OpenCV library.

fisheye image and the cropped satellite image (cropped using GPS initialization, refer Figure 7.4). The matched features are used to solve a Perspective-n-Point (PnP) problem [90, 50] to estimate the initial pose of camera (also called the PnP estimate) in the prior map reference frame $\mathbf{W}$. As

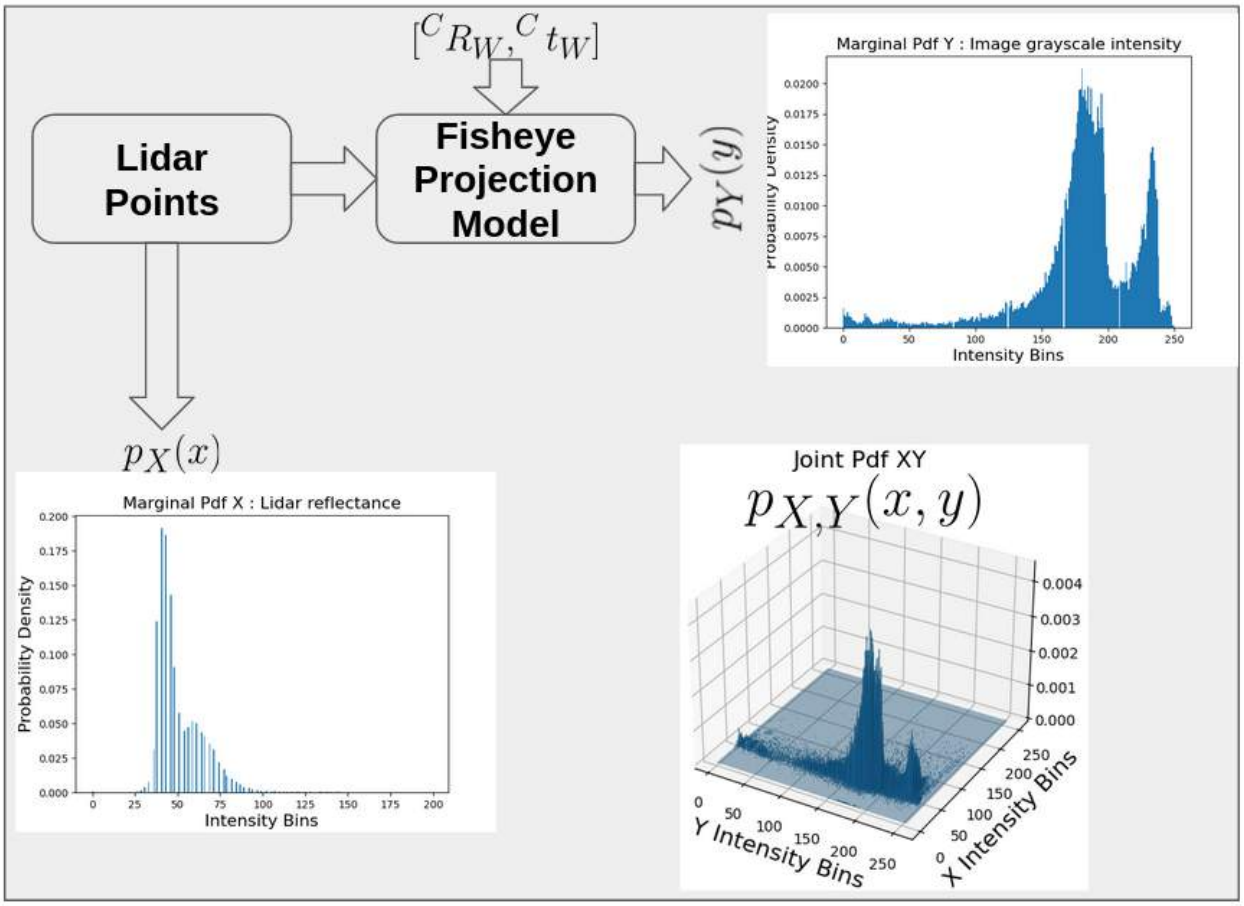


Figure 7.7: The process of MI maximization involves calculation of marginal and joint propbability distributions from intensity histograms from both fisheye camera and LiDAR mapping data. Next, individual and joint entropies are estimated which is used to calculate mutual information (MI). The MI needs to be maximized between LiDAR intensties, and pixel intensities at LiDAR point projections on the fisheye Image to arrive at a refined estimate of $[R_C^W, t_C^W]$. The initial camera localization from Section 7.6.1 is used to initiate a grid search around the initialization point and search for registration parameters that maximizes MI.

the metric scale of the satellite image (1 pixel = 0.1 m) is known, the camera pose is obtained in metric units. A schematic of this process is presented in Figure 7.6.

7.6.2 Refinement of camera localization using Maximization of Mutual Information

Mutual Information (MI) has been used in several fields for registering data from multi-modal sensors [91, 92]. The initial camera pose estimate from Section 7.6.1 is refined by maximizing the Mutual Information (MI) between the LiDAR reflectivity of ground points and the fisheye

grayscale values at the pixel locations onto which the LiDAR points are projected using the camera pose $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$ (Figure 7.7).

7.6.2.1 Theory

MI (Equation 7.2) provides a way to statistically measure mutual dependence between two random variables X and Y.

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \tag{7.2}$$

Where $H(X)$ and $H(Y)$ are the Shannon entropy over random variables X and Y respectively, and $H(X,Y)$ is the joint Shannon entropy over the two random variables:

$$H(X) = -\sum_{x \in X} p_X(x) \log p_X(x) \tag{7.3}$$

$$H(Y) = -\sum_{y \in X} p_Y(y) \log p_Y(y) \tag{7.4}$$

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x,y) \log p_{XY}(x,y) \tag{7.5}$$

The entropy $H(X)$ of a random variable X denotes the amount of uncertainty in X, whereas $H(X,Y)$ is the amount of uncertainty when the random variables X and Y are co-observed. The formulation of MI in Equation 7.2 shows that maximization of $MI(X,Y)$ is achieved by minimization of the joint entropy $H(X,Y)$, which coincides with minimization of dispersion of two random variable's joint histogram. The process of estimation of marginal and joint probabilities is described in Section 7.6.2.2 and pictorially shown in Figure 7.7.

7.6.2.2 Mathematical Formulation

Let $\{\mathbf{P}_{\mathbf{W_i}}; i = 1, 2, \ldots, n\}$ be the set of 3D points whose coordinates are known in the prior map reference frame $\mathbf{W}$ and let $\{X_i; i = 1, 2, \ldots, n\}$ be the corresponding reflectivity values for these points ($X_i \in [0, 255]$). Equation 7.1 presents the relationship between $\mathbf{P}_{\mathbf{W_i}}$ and its image projection $\mathbf{p_i}$ as a function of $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$. Let $\{Y_i; i = 1, 2, \ldots, n\}$ be the grayscale intensity of

the pixels $\mathbf{p_i}$ where $\mathbf{P_{W_i}}$ project onto, such that:

$$Y_i = I(\mathbf{p_i}) \tag{7.6}$$

where $Y_i \in [0, 255]$ and I is the grayscale fisheye image. Therefore, X_i is an observation of the random variable X, and for a given $[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$, Y_i is an observation of random variable Y. The marginal ($p_X(x)$, $p_Y(y)$) and joint ($p_{X,Y}(x, y)$) probabilities of the random variables X and Y, required for calculating MI (Equation 7.2), can be estimated using a normalized histogram (Equation 7.7):

$$\hat{p}(X = k) = \frac{x_k}{n}, k \in [0, 255] \tag{7.7}$$

where x_k is the observed counts of the intensity value k.

7.6.2.3 Global Optimization

${}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}} \in SO(3)$ is an orthonormal rotation matrix which can be parameterized as Euler angles $[\phi, \theta, \psi]^\top$ and ${}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}} = [x, y, z]^\top$ is an Euclidean 3-vector. ψ is the rotation of the camera along its principal axis. In the current context, the fisheye camera is facing vertically downward so it is not necessary to refine the ϕ & θ and leave it at what the feature matching based technique (Section 7.6.1) determines it to be, which is very close to 0. Therefore, as far as rotation variables are concerned, only ψ is refined. The variables optimized together are $\Theta = [x, y, z, \psi]^\top$. The optimization is posed as a maximization problem:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, MI(X, Y; \Theta) \tag{7.8}$$

7.7 Experiments And Results

This section describes the experiments performed to evaluate the proposed technique using data obtained from both simulator and real world sensors.

7.7.1 Simulation Studies

The approach is first evaluated on a simulator which is built using data from real sensors. The Mathworks' tool RoadRunner [93] is used to generate the 2D features like lane geometry and lane markings with the satellite map used as a reference. The 3D structures are created using the Unreal Engine Editor [94] with the help of real satellite and 3D LiDAR maps. Since the simulated environment is created using the prior map components, it can be safely assumed that the simulator aligns with the real world to a high degree of accuracy. In order to generate the fisheye images, a fisheye camera is modelled in Unreal Engine using the equidistant model with a field of view of 180 degrees. The approach has been demonstrated in simulation for the fisheye image shown in Figure 7.2a. The fisheye image is first rectified (Section 7.5.1.2) to generate Figure 7.2b, which is used for estimating the initial camera pose using the approach in Section 7.6.1. Next, the initialization is refined using maximization of MI (Section 7.6.2).

The camera registration estimate ($[{}^{\mathbf{C}}\mathbf{R}_{\mathbf{W}}, {}^{\mathbf{C}}\mathbf{t}_{\mathbf{W}}]$) is qualitatively validated (Figure 7.10) by projecting points from the 3D-LiDAR map (Figures 7.3b and 7.3c) onto the fisheye image (Figure 7.2a). As shown in Figure 7.10a, the projection of LiDAR map points on the fisheye image obtained using the initial camera pose are not well aligned. When the MI around the initial camera pose (Figure 7.8) is plotted, it is observed that it is not at its maximum at the initial estimate (also called PnP Estimate), thus holding the promise for further improvement. Similarly, Figure 7.9 presents the surface plot of MI, which shows the presence of a global maximum in each subplot. Therefore, on solving the optimization problem posed in Equation 7.8 one obtains the pose which maximizes the MI between the two modalities and results in negligible misalignment of the projected LiDAR map points in Figure 7.10b.

100 independent trials are run to evaluate the robustness of the MI based refinement method (Section 7.6.2) to change in initialization (Figure 7.11). The translation parameters show greater variance when compared to yaw ψ. The higher variance of the translation variables can be attributed to the fact that the MI based cost function is less sensitive to changes in translation variables, especially in the outdoor scenario where most of the points lie in the far field. In the limiting

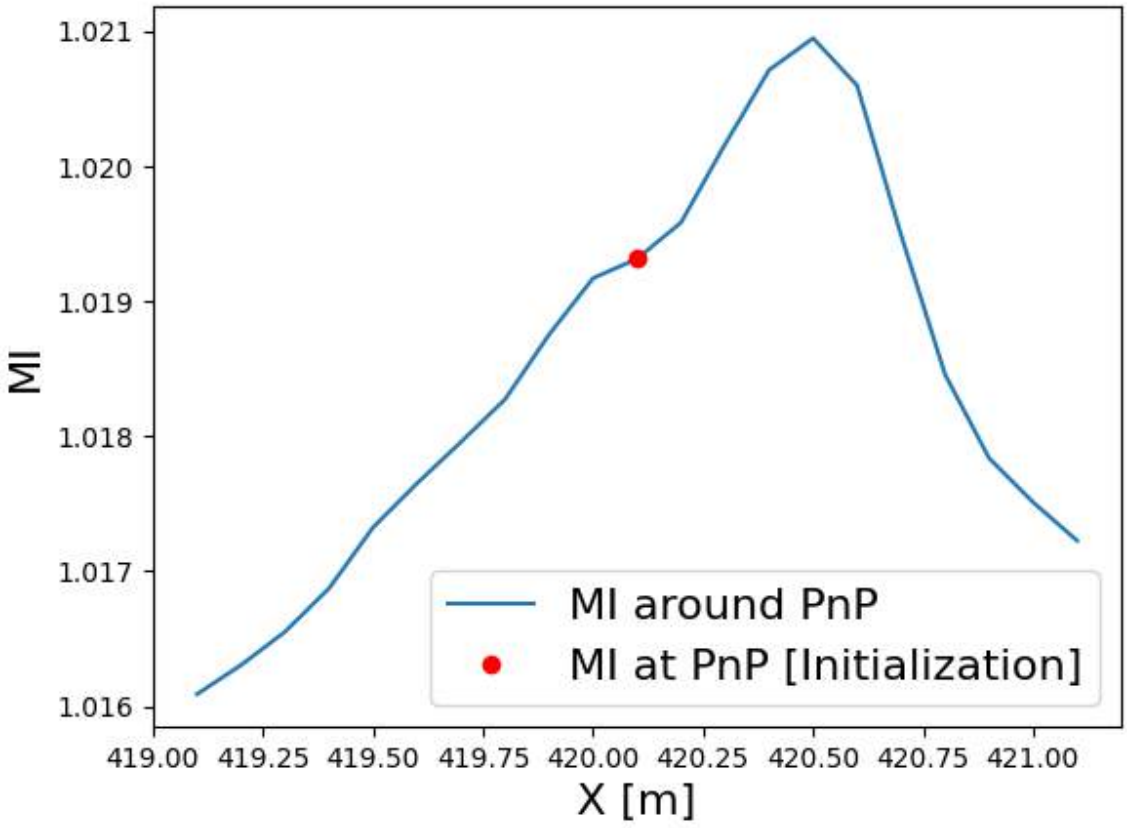


Figure 7.8: Plot of MI around the PnP estimate (from Section 7.6.1). Plot shows that MI is not at maximum at the PnP estimate, therefore the maximization of MI may reduce the misalignment in projection of 3D-LiDAR points visible in Figure 7.10a

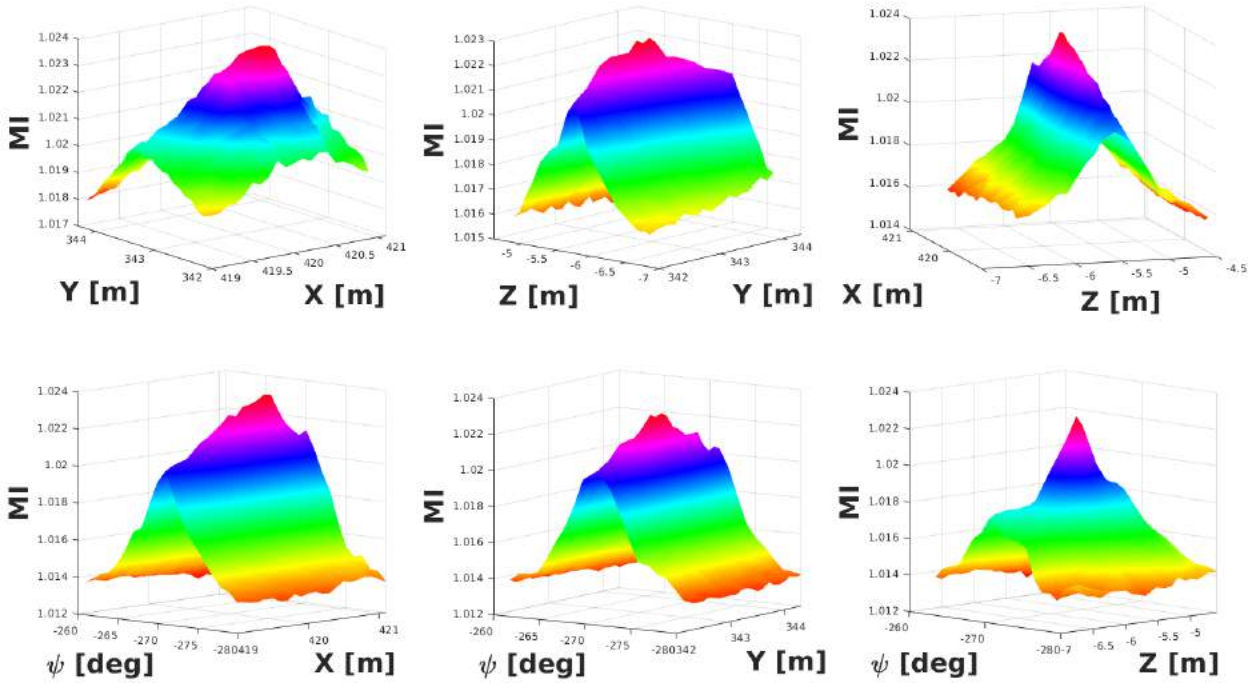
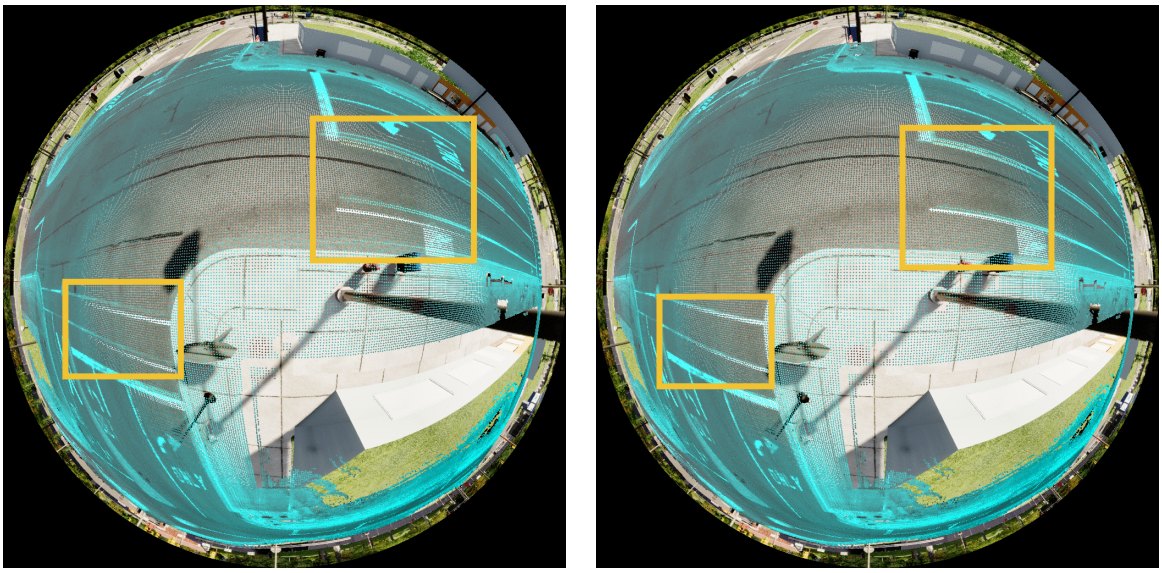


Figure 7.9: **Plot of MI by perturbing two degrees of freedom around the PnP Estimate** (i.e. the initial estimate from Section 7.6.1). The cost function from single Image LiDAR Scan pair is not differentiable at several points. Hence, an exhaustive grid search around the initialization point has been used to arrive at solution where MI is maximized.



(a) (b)

Figure 7.10: Two step approach for camera localization in prior map : Figure 7.10a shows the projection of 3D-LiDAR ground points (cyan) using the initial estimate obtained using feature matching (PnP Estimate from Section 7.6.1), and Figure 7.10b shows the projection of 3D-LiDAR ground points using the refined camera pose estimate from maximization of MI (from Section 7.6.2). The misalignment visible in Figure 7.10a, is absent in Figure 7.10b (best viewed digitally).

case, far away points are considered to be points at infinity, represented as $[x, y, z, 0]^T$, which, under camera projection (Equation 7.1), render the translation variable (${}^{C}\mathbf{t_W}$) in the optimization problem (Equation 7.8) un-observable. This result is also presented in [27], specifically when discussing sensor registration in an outdoor environment using only a single image - LiDAR scan pair, which is similar to the situation presented here.

7.7.2 Real World Experiments

Experiments were performed with data collected from a real fisheye camera (Figure 7.12) in order to demonstrate the validity of our algorithm in realistic situations.

7.7.2.1 System Description

A fisheye lens Fujinon FE185C057HA 2/3 inch sensor, which provides 185° of vertical and horizontal FoV is used for real world experiments. The camera is a 5MP Sony IMX264. The

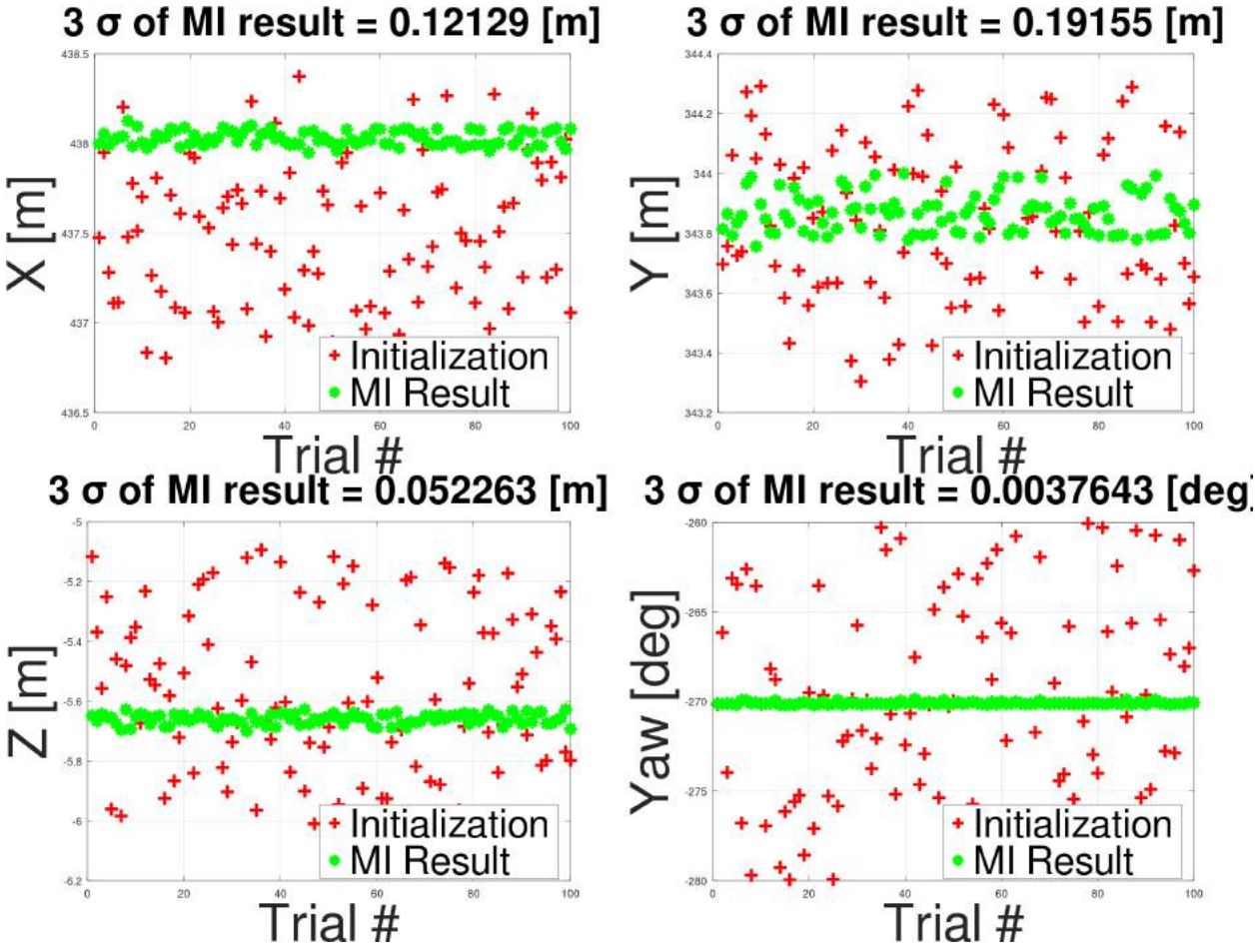


Figure 7.11: Performance of MI based fisheye camera localization refinement for different initial conditions. Here 100 independent trials are performed. + is the initialization, * is the final result.

sensor is mounted from a tripod (Figure 7.12), looking vertically down, and capturing images of the environment. The ultimate goal is to mount these cameras at challenging intersections for navigation of autonomous vehicles, and use the proposed method to register them in a prior map. An iPhone has been used to provide approximate GPS location (without the orientation) of the fisheye camera, which is used to limit the search space in the prior map. A high accuracy RTK-GPS (uBlox ZED F9P GNSS + uBlox antenna ANN-MB-00) unit has been used to measure GPS co-ordinates of distinctive corners on road markings that can be used for quantifying the accuracy of camera localization (Figure 7.14).

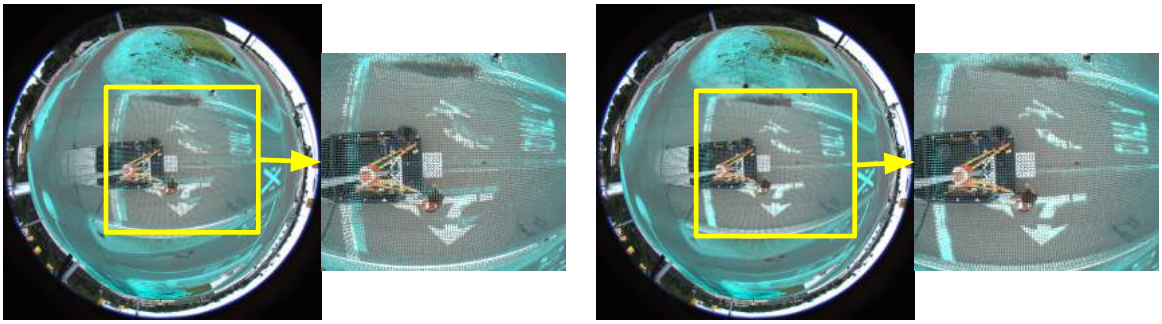
7.7.2.2 Results

Results with real world data are presented from two different locations in Figure 7.13 which qualitatively demonstrate the incremental improvement in camera localization using the two step registration approach. While the projection of LiDAR points onto the fisheye image using the initial camera localization appears misaligned in Figures 7.13a and 7.13c, the misalignment is reduced when the LiDAR points are projected using refined camera localization in Figures 7.13b and 7.13d. The veracity of camera registration has been quantified by measuring the average re-projection error for points on the fisheye image whose GPS locations have been measured using high accuracy RTK-GPS. These points on the fisheye image have been manually marked, and the difference between them and the reprojection of the corresponding 3D point in the prior map onto the fisheye image have been measured using the estimated camera localization. Since the reprojection error is a function of the estimated camera registration pose, a lower reprojection error may imply a better camera pose estimate. The results presented in Figure 7.14 show that the reprojection error on the fisheye image plane reduces when the initial camera localization is refined by maximization of mutual information.

7.8 Discussion

This work presented an approach to register a smart infrastructure node equipped with a fisheye camera. The downward facing fisheye image is registered to a prior map, comprising of a co-

Figure 7.12: Collecting data for real experiments using a tripod mounted downward looking fisheye camera. The ultimate goal is to mount these cameras, along with our smart infrastructure nodes, at challenging intersections for navigation of autonomous vehicles.



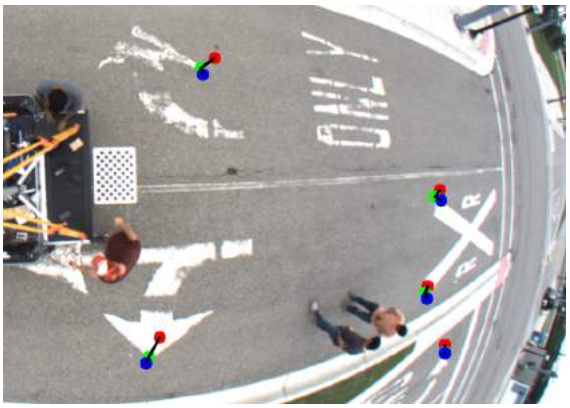
(a) Location 1 - Projection of LiDAR points (cyan) using PnP Estimate

(b) Location 1 - Projection of LiDAR points (cyan) using MI Estimate

(c) Location 2 - Projection of LiDAR points (cyan) using PnP Estimate

(d) Location 2 - Projection of LiDAR points (cyan) using MI Estimate

Figure 7.13: **Real World Experiments:** Projection of 3D-LiDAR ground points (cyan) on to fisheye camera using the initial camera pose (PnP Estimate) (Figure 7.13a, 7.13c) and MI based refinement of initial camera pose (Figure 7.13b, 7.13d). The misalignment of LiDAR points visible in highlighted areas in Figures 7.13a, 7.13c, are minimized in Figures 7.13b, 7.13d



(a) Location1 - Average Reprojection Error with PnP = 20.52 pixel, and with Maximization of MI = 9.12 pixel

(b) Location2 - Average Reprojection Error with PnP = 26.40 pixel, and with Maximization of MI = 13.11 pixel

Figure 7.14: Green Circle - Hand annotated corner point whose GPS location was measured using a high accuracy RTK-GPS unit, Red Circle - Projection of corner point's position onto fisheye Image using PnP estimate (Section 7.6.1), Blue Circle - Projection of corner point's position onto fisheye Image using MI estimate (Section 7.6.2).

registered satellite image and a ground reflectivity+height map from LiDAR-SLAM. The two-step approach uses feature matching between the rectified fisheye image and the satellite imagery to obtain an initial camera pose, followed by maximization of MI between the fisheye image and 3D LiDAR map to refine the initial camera registration. Since only a single camera image and LiDAR scan are used for sensor pair registration (the smart infrastructure node is static), the cost surface may not always be smooth [27], and therefore not differentiable - leading to the failure of gradient descent methods. Hence, an exhaustive grid search method apprach is utilized to find the optimal camera pose. Such a search may be time-consuming (depending on the number of 3D LiDAR map points used for calculating MI (Equation 7.2), the interval of exhaustive grid search and the available compute power), and not suitable for real-time operation. This is acceptable for the given application, because one needs to localize the smart infrastructure node once at install or during maintainance, which can be an offline process. Moreover, this method can be accelerated by use of GPUs. A commonly raised question is if the fisheye camera can be registered in the prior map using a GPS receiver? Although, a GPS receiver can localize the camera with some degree of

accuracy it still cannot report an accurate estimate of height unless a highly accurate and expensive GPS receiver is employed, moreover a simple GPS receiver cannot provide orientation. Lastly, inorder to use the GPS to register the fisheye camera in a prior map one also needs to register the GPS receiver with respect to the fisheye camera which will prove to be an onerous task in itself.

7.9 Conclusion

The current Chapter took a departure from the standard approach taken in the previous Chapters to register sensors. While the previous Chapters aligned only sparse features extracted from sensor data, the current Chapter aligned dense sensor data by transforming sensor readings into normalized probability distributions. The dense alignment of probability distributions was initialized by a sparse feature-matching technique. The sparse feature matching technique cannot provide the best value of sensor registration as it was applied on rectified fisheye image unlike the maximization of mutual information which was done on the fisheye image space. Moreover it is difficult to match features like pixel patches and lines between fisheye image and LiDAR intensity data because a straight line in LiDAR scan will not look like a straight line in fisheye image owing to distortions from the fisheye projection model.

Irrespective of the physical quantities aligned, the registration technique in this Chapter is in principle similar to the approaches presented in the previous Chapters where data from different sensors are used to estimate pose of one sensor with respect to the other. The approach taken in the current Chapter may be used in a multi-sensor setting in addition to sparse feature alignment method to perform registration of several sensors together.

CONCLUSION

8. CONCLUSION AND FUTURE DIRECTIONS

8.1 Summary of the Thesis

This thesis presents several methods of performing multi-sensor registration of robotic sensors, *viz.* Cameras, IMUs and LiDARs, commonly used by autonomous agents for perception, state estimation, planning and navigation. The cross calibration problem for sensor systems can be sub-categorized into extrinsic and temporal calibration, and the focus of this thesis is primarily the former. The thesis presented novel techniques to perform pair-wise calibration of Camera-LiDAR system (Chapter 2), LiDAR-IMU system (Chapter 3) and also presented an open-sourced implementation [1] of [8] for Camera-IMU systems (Chapter 4). The ideas described in Chapters 2, 3, 4, used for pair wise calibration of sensor pair, are utilized together in Chapter 5 for joint calibration of Camera-IMU-LiDAR system which to the best of the author's knowledge is the only approach in literature as far as joint extrinsic calibration of these sensors is considered.

The Camera-LiDAR pair-wise calibration method presented in Chapter 2 describes a novel target-based calibration method using the back-projected plane of an edge detected in camera (Section 2.4.3.2) for constraining the calibration optimization problem, this technique performed better when compared against two other target based calibration methods [24] & [34]. Moreover, the detection of edges of the calibration target helps to quantify the quality of sensor extrinsic calibration by means of a metric called the mean line re-projection error (MLRE) (Section 2.7.3). The advantage of using such a metric in the context of Chapter 2 is that the verification metric is an independent one, and was not minimized in the optimization process. The limitation of this verification approach is that it quantifies the quality of extrinsic calibration which is a 3D-pose in 2D space (by measuring reprojection errors). Another limitation of this work is that the method does not perform temporal calibration of the Camera-LiDAR system. In fact, the author has not come across any contribution which performs temporal calibration of a Camera LiDAR sensor pair using appearance based techniques, so this still remains an area of future contribution.

[1] `https://github.com/unmannedlab/camera_imu_calibration`

The LiDAR-IMU pair-wise calibration method presented in Chapter 3 is an EKF based extrinsic calibration method which derives motivation from Open-Vins [8] and [57] for using the EKF as the estimation engine, and also from [18] for using the motion based calibration approach as EKF update step for pair-wise calibration. The proposed method differs from other LiDAR-IMU calibration methods [1] & [16] in the manner that the later both use Gaussian process regression and continuous time spline based interpolation methods respectively for performing extrinsic calibration. These batch optimization methods will be difficult to extend to online application whereas the EKF based framework can be easily extended to online operation. Moreover both of these methods depend on presense of dominant planar structures in the environment for performing LiDAR-IMU calibration, the proposed method doesn't have any such stringent requirements. The author has also open sourced [2] the implementation of the proposed LiDAR-IMU extrinsic calibration method for the wider robotics community. After [16] this is the only open-sourced LiDAR-IMU calibration tool box to the best of author's knowledge. However, the proposed method does not estimate temporal calibration as [1] & [16] do. As far as testing the veracity of the calibration estimate is concerned, a relative verification technique has been used which does not require an auxilliary sensor like a camera for verifying the result of extrinsic calibration. The idea is to collect calibration data in one configuration then move one of the sensors by a known amount and again collect calibration data with the moved sensor at its new displaced location. Next, the calibration algorithm should be run on both of these datasets and the difference of the resulting estimates must be compared against the known displacement. The advantage of using relative verification is that it doesn't require the use of an auxilliary sensor and the verification metric is an error in 3D (or in $SE(3)$ if one chooses to represent it as such) - the same dimension as the calibration estimate. Although relative verification gives us an idea about how well the calibration algorithm works, it is not feasible to count on such a technique in real situations where sensors are not expected to be disturbed just for verifying the result of registration. The possible solution is to use another LiDAR or Camera as a verification sensor and the downside to that is that one needs to calibrate

[2] `https://github.com/unmannedlab/imu_lidar_calibration`

the auxilliary sensor with respect to the already exisiting sensor pair. Moreover, if a camera is used as an auxilliary verification sensor in LiDAR-IMU calibration then the verification metric will not only be in 2D image space but also be only qualitative, as it involves projection of LiDAR points on camera image for visually verifying how good the projection is.

As far as Camera-IMU pairwise extrinsic calibration is considered (Chapter 4), this thesis makes three contributions to the field. First, the thesis provides an open sourced implementation of the calibration technique presented in publication [8]. This will provide an easy alternative to the popular open-sourced camera IMU calibration tool-box called Kalibr [7] which happens to be the only open sourced camera IMU tool available to the robotics community. Moreover, Kalibr is getting cumbersome to work with owing to compatibility issues with latest versions of Ubuntu and Python. Second, this self implementation of [8] has been extensively compared against Kalibr (Section 4.9.3) with the conclusion that the performance is comparable or similar. Lastly, the self implementation of [8] (called RepErrCal) has been compared against the case when the camera is used as a pose sensor (called MoCal) as done in Chapter 3 for LiDAR-IMU calibration (where the LiDAR has been used as a pose sensor). The conclusion is that the quality of calibration estimates is not good with MoCal and therefore RepErrCal is a better approach for performing Camera-IMU pairwise calibration (Section 4.9.4). Chapter 4 also motivates the author to use the reprojection error minimization metric as done in [8] for Camera-IMU update in the joint calibration of LiDAR-IMU-Camera instead of using the motion based calibration constraint as done in Chapter 3 and [18]. As far as verification of Camera-IMU calibration is concerned, it's done by measuring the re-projection error between detected checkerboard corners and projected pixels of known 3D positions of the checkerboard corners. As mentioned previously, this way of verifying calibration is not ideal as the verification metric is in a 2D image plane but this is the best one can do without usage of an auxiliary sensor or resorting to relative verification techniques.

LiDAR-IMU-Camera joint calibration has been presented in Chapter 5 which incorporates the ideas presented in Chapters 2, 3, 4 for joint calibration of these sensors. The camera IMU calibration resulting from this joint estimation process is found to be close to the results reported

by individual camera-IMU calibration using self implementation of [8] (called RepErrCal). But, the LiDAR-IMU calibration from the joint estimation routine shows improvement over individual pair-wise calibration of LiDAR-IMU system as evident from qualitative evaluation using projection of LiDAR points on camera image (Figure 5.5).

In addition to the aforementioned techniques which involves a mobile sensor suite to be mounted on a robot or any autonomous agent for onboard data acquisition for perception, planning & state estimation, the thesis also presents sensor registration approaches relevant to smart infrastructure based sensing for autonomous vehicles. Infrastructure based sensing is relevant in scenarios involving complex traffic intersections where an autonomous vehicle cannot see around the corner or in scenarios when the sensors onboard the autonomous vehicle fail to operate. However, these smart infrastructure sensors need to be registered in the map which the autonomous vehicles use so that the data from them can be represented in the same coordinate frame of reference as the (map) frame in which the autonomous vehicle operates.

An area where smart infrastructure based sensing is gaining ground is in controlled environments like factory floors and parking lots for autonomous operation of unmanned vehicles. These cameras may or may not share any field of view and can be used for solving the localization problem in controlled settings, aiding autonomous parking, fork-lifting or haulage. In order to make this possible the network of cameras must be registered with respect to a common frame of reference so that the knowledge of the pose of a camera in the network can also provide one the knowledge of the pose of a vehicle detected in the same camera. This information can be used for motion planning, control and downstream perception tasks. To this end Chapter 6 of the thesis presents an approach to register approximate 40 cameras spread over a large area in an indoor parking facility. A calibration robot is driven under the network of infrastructure cameras and a two way detection, i.e. detection of a infrastructure camera in calibration robot's sensor and vice-versa is used to constrain a large joint optimization problem which simultaneously estimates the pose of the infrastructure cameras, the pose of the calibration robot and also a map of visual features in the environment. As this approach also estimates the calibration robot pose, it can be used for local-

ization of a vehicle driving under the field of view of the network of infrastructure cameras. The pose estimates of each infrastructure camera in the large network were compared against camera pose estimated from a similar approach which used LiDAR Odometry instead of Visual Odometry for calibration robot pose estimation (Section 6.7.2).

In Chapter 7, the thesis presents a two stepped approach to register a static, elevated, downward looking, wide angled fisheye camera in a prior map built from combination of satellite imagery and LiDAR mapping data. Once the fisheye camera is registered it will generate measurements which can be represented in the same frame of reference as the autonomous vehicle and under certain circumstances has the potential to aid autonomous driving without using any onboard sensing. In this approach, first feature-matching is performed between rectified fisheye image and metric satellite image to initialize the fisheye camera pose in the prior map (Section 7.6.1), next the mutual information between fisheye image and LiDAR map is maximized inorder to refine the initialized camera pose (Section 7.6.2). Since the mutual information cost function is calculated using only a single fisheye image LiDAR map/scan pair, the surface of the cost function is not smooth which makes application of gradient based methods difficult. Therefore grid search based optimization is used to register the fisheye camera pose in the prior map. As far as verification of registration is concerned, the GPS location of few points on the road are recorded using a high precision RTK-GPS, then these points are projected onto fisheye image using the final estimate of camera registration. The projection of the GPS measured points are compared against manual annotation of the corresponding points on the fisheye image and the re-projection error is used as a measure of verification (Section 7.7.2). The points are manually annotated on the fisheye image, so there may be some errors in that process as well. As discussed previously, the verification technique here involves calculation of re-projection error in the 2D image pixel space where as the quantity being estimated is a 3D pose, this has been a recurring limitation in verification of virtually all registration/calibration methods which do not use an auxiliary sensor or relative verification for validation purposes - both of which have their own shortcomings.

8.2 Key Insights

The most significant take away from all registration algorithms is that the quality of results depends on the nature of calibration data fed to the algorithm. Consequently, calibration data collection is one of the most important steps in sensor registration. Calibration data (in terms of performing calibration motion manuevers or collecting diverse environment data from exteroceptive sensors) is either collected by exciting the sensor suite or changing the environment of the sensors. There are contributions in literature which talk about how much variation in calibration data is necessary to ensure full observability and reliable convergence of calibration data, but practical wisdom has usually been that one must collect as much diverse calibration data as possible subjected to practical constraints and limitations. It is not always easy to ascertain that the calibration data will ensure full observability owing to physical and practical limitations. Although the sensor systems used in Chapters 2 - 5 are such that they share fields of view and can be motion excited about and along all rotational and translational axes respectively owing to the fact that the sensor suite is handy, such convenience cannot always be guaranteed in many real world settings, e.g. full motion excitation of an autonomous vehicle cannot be ensured, and also sensors may be far apart or looking in different directions thus not sharing any common field of view.

As far as calibration of a pair of exteroceptive sensors is concerned, they need to share fields of view if appearance based registration is being performed. If appearance based target-less registration is being performed then one may also need to solve the problem of data association between the two exteroceptive sensors, which may not be cumbersome if the sensors are of same modality but is a major challenge when sensors of different modalities are used. However, if the exteroceptive sensors are calibrated using motion based calibration technique then one needs to estimate individual sensor motion (LiDAR or Visual Odometry as the case may be) and align corresponding motion segments by matching timestamps - which may further necessiate temporal calibration. The exteroceptive sensors need not share the same field of view for motion based calibration. The limitation of the motion based registration method is that for most cases, especially in the case of autonomous ground vehicles, the motion excitation during calibration data collection will be

limited to only 3 DoF (2 translation , 1 rotation) which will limit the observability of the full 6 DoF extrinsic calibration between the sensors being calibrated. If the exteroceptive sensors share common field of view, then this limitation may be overcome by using heuristics or by aligning data from both sensors which correspond to the same class in both sensors, eg. the ground plane. This will require an additional segmentation/classification layer over raw sensor data. Alignment of corresponding data between two sensors may help recover/observe the unobservable 3 degrees of freedom of the 6 DoF pose. The advantage of motion based registration approach is that it is more intuitive to perform temporal calibration under a motion alignment framework - alignment of rotation motion only may help estimate the unknown time calibration between sensors.

If a sensor system consists of a proprioceptive sensor like a wheel encoder or an IMU then the system definitely needs to be motion excited while collecting calibration data because motion is the only physical quantity that proprioceptive sensors can measure. The exteroceptive sensors in such a system need to do frame to frame motion estimation by tracking features or full sensor data alignment between frames. The challenge in this case is that since proprioceptive sensors do not sense the environment, no data association can be attempted between an exteroceptive sensor data and a proprioceptive sensor data, making it difficult to estimate unobservable degrees of freedom in case of physical limitation to perform full 6 DoF motion excitation while collecting calibration data. However, under regular driving in on-road urban situations the authors have measured $\pm 10^{\circ}$ vehicle roll/pitch from LiDAR IMU SLAM system. So the goal in the future is to verify if these small rotations are enough to observe the otherwise unobservable degrees of freedom.

As far as camera-LiDAR calibration in Chapter 2 is concerned, the author had to ensure that the edges of the calibration target can be easily detected in both camera $\&$ LiDAR while collecting data for Camera-LiDAR registration. As far as the camera is concerned it had to be ensured that there is enough contrast between the calibration target and the background so that the edges can be easily detected, similarly in the case of LiDAR it had to be kept in mind that the calibration target should be held diagonally up/down so that all four edges of the square target could be detected in the LiDAR sensor. The detection in VLP-32 LiDAR is more tedious because the distribution of

LiDAR channels follows a Gaussian distribution which is thick in the center and thinner towards the edges, hence the calibration board should be held at a height which coincides with the center of the LiDAR sensor. Such limitations constrain the range of motion one can exert to the calibration board inorder to obtain diverse views in the calibration data necessary for ensuring observability of camera registration.

As far as LiDAR-IMU (Chapter 3), Camera-IMU (Chapter 4) and LiDAR-IMU-Camera (Chapter 5) registration is concerned, it was relatively simpler to collect calibration data because the sensor suite installed on the experimental robot has all sensors located closely on a single rigid body (Figures 2.1, 3.1, 5.1) which makes it handy and easy to move. However, it was not possible for the author to use the Point to Back-Projected Plane (used for Camera-LiDAR calibration in Chapter 2) constraint between LiDAR edge point and corresponding back-projected plane in camera while performing joint calibration of LiDAR-IMU-Camera system because of the difficulty in detecting calibration board edges in both camera and LiDAR as described in the previous paragraph. When only LiDAR-Camera system was being calibrated, only moving the calibration board was necessary, but with LiDAR-IMU-Camera system, as IMU is a proprioceptive sensor, the sensor suite needs to be motion excited. The motion excitation of the sensor suite makes detection of the edges of the calibration target more challenging.

Another recurring issue with most registration/calibration algorithms is the difficulty with metric evaluation of the calibration result. If the sensor suite consists of a camera then the most preferred method of verifying the result of calibration is either the calculation of re-projection errors (in camera-camera $\&$ camera-IMU systems), or qualitative visual verification by projection of LiDAR points on camera image plane (in camera-LiDAR $\&$ LiDAR-IMU-Camera systems). Re-projection error is calculated on 2D image pixel space and cannot be reliably used to verify a 3D quantity (the calibration estimate) because the projection on 2D image plane results in loss of dimension, and a point moving along a ray emanating from camera center will project at the same point on the image plane, irrespective of its position on the ray. Because of the 2D projective geometry, re-projection error also varies on changing the point of view. So, a low re-projection

error on a single image frame does not necessarily mean that the estimated calibration is correct. That is why the reprojection error is averaged over several frames with varying view points.

If it's a LiDAR-IMU system then relative verification methods or CAD references can be used for verifying the result. However, CAD models are difficult to make if the sensor and the robot platform are all procured from different sources and assembled together. Besides, CAD models will not tell one the exact origin or center of a camera sensor. Moreover, in practical lab scenarios researchers may often change the position of sensors making reliance on CAD models impractical. Relative verification technique discussed in Chapter 3 is used for verifying if the calibration algorithm is converging to the correct value by displacing sensors by known amount and re-performing the calibration routine. Although the method tells the user if the algorithm is converging close to the correct calibration estimate or not, practically it is not suitable as it is not expected to displace sensors on an system just to check the veracity of calibration result. To conclude, it is better to use a cheap aiding/auxilliary sensor like a camera for verifying LiDAR-IMU registration and improve it further by using the joint calibration process described in Chapter 5.

8.3 Future directions

The immediate extensions to the work presented in this thesis for all the registration scenarios is the incorporation of temporal calibration between the concerned sensors. Temporal calibration in the appearance based (both with and without calibration target) registration of camera and LiDAR is going to be a novel contribution as the author has not come across any appearance based camera LiDAR registration algorithm that estimates the temporal offset between the sensors. This is going to be future work for the camera LiDAR extrinsic calibration present in Chapter 2. As far the EKF based sensor registration algorithms presented in Chapters 3 and 5 for LiDAR-IMU and LiDAR-IMU-Camera respectively is considered, the immediate goal is to follow the temporal calibration method presented in [95] where the time offset between two sensors is included in the EKF state vector for estimating it concurrently with all other states of interest. Finally, for the multi camera registration algorithm presented in Chapter 6, a continous time approach (B-splines, Gaussian Process Regression) needs to be adopted for modelling the system in-order to estimate individual

system time offset w.r.t to a reference sensor.

In addition to temporal calibration, another direction of future work is the study of observability of registration parameters for all the registration algorithms addressed in this thesis. The focus in this thesis has been to collect enough diverse data to ensure observability but the property of parameter observability requires more rigorous mathematical treatment which will have far reaching impact on implementation of algorithms that can perform long term calibration monitoring and correction under a SLAM framework. Long term calibration under SLAM is necessary because sensors may undergo physical displacement during the course of motion in an autonomous robot. Although such physical displacement will be small in urban on road settings in short term but long term wear and tear and mechanical vibrations can result in significant change in calibration parameters in the longer term. Calibration parameter monitoring and correction become more important in off road wheeled robots and in regular legged robots because in these cases the robotic platform experiences more mechanical vibrations. But one also needs to keep it in mind that if the platform moves and vibrates more then the data measured by the sensors may be more suitable for peforming calibration as all degrees of freedom may become observable owing to the diversity of data collected. Study of observability is important for long term calibration and monitoring because such an analysis can tell the user or the software stack which among the calibration parameters can be monitored and corrected given the measurements subjected to physical limitations of the mobile platform. The user would then expend their energy to track and correct the observable parameters only and make sure that the unobservable parameters are set to the correct value during routine maintainace. The EKF framework discussed in Chapters 2 - 5 is suitable for online calibration monitoring and correction as it is a recursive filtering approach which tracks and updates parameters on reception of new sensing data. The χ^2 test within EKF framework can inform the software/user about anomalies in the spatial and temporal alignment of measurements from the sensors which can be used to detect change in the calibration due to effects of the physical environment, and correct the calibration estimate.

Another aspect that needs attention is to solve the problem of data association across sensing

modalities which can ease the problem of feature association between sensors like cameras & LiDARs for performing appearance based sensor registration. Deep learning techniques can come handy in this application. Moreover, if the problem in associating features across sensors is addressed then it can not only be applied to appearance based registration methods but under certain circumstances can be used to solve the problem of observability discussed in the previous paragraph (For LiDAR camera registration in a self driving car, one can do motion based allignment of x, y & yaw, and appearance based alignment of z, roll & pitch).

Most important of all, in conclusion, all registration algorithms need universal or absolute methods to validate the calibration estimates in 3D ($\in SE(3)$) because the sensor registration pose estimate is a physical quantity in $SE(3)$. Whenever a camera sensor is involved the convenient and prevalent way is to measure the 2D reprojection error of known 3D positions and as discussed earlier reprojection error is a scalar error in 2D image plane, and it cannot completely capture the veracity of a quantity in $SE(3)$. Finally, the most important question is how good a the registration estimate should be? To which the author belives the answer is, it depends. If the application involves space operations or performing high precision robotics like robotics surgery, precision manufacture etc, then one needs to have the best possible value of calibration parameters because the room to commit errors is minimal. In other cases, where error tolerances are higher, reactive planning methods can handle local limitations in sensing and state estimation that may stem from less accurate calibration parameters.

8.4 Final Thoughts

8.4.1 Generalized Sensor Registration

From the sensor registration methods implemented in the previous Chapters, it can be concluded that any registration problem is ultimately a sensor pose estimation problem where the pose of all the concerned sensors moving or static are estimated as a result of the registration procedure using measurements made by the sensors themselves. The sensor poses to be estimated form unknown variables in measurement residuals formed by known measurements and unknown sensor

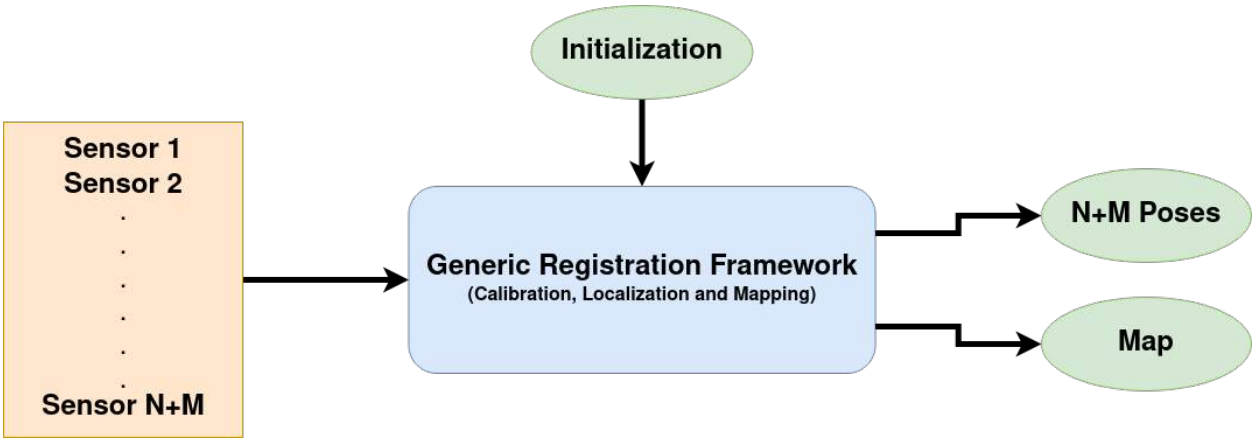


Figure 8.1: Generic Registration Framework

poses. These variables may be estimated by filtering or batch optimization techniques. Depending on the sensors used, maps may also be estimated, although the type of map depends on the sensors being registered. Consider a scenario which has M static sensors in the environment and N sensors mounted on a sensor suite rigidly attached to a mobile robot, in such a situation a multi sensor registration algorithm should estimate the static poses of M static sensors, the evolving pose of the sensor suite on the mobile robot, i.e. the evolving pose of one of the sensors called base sensor on the sensor suite, and also the static extrinsic calibration of N-1 sensors w.r.t the base sensor. The registration algorithm should also generate a map of the environment. In the described scenario, the registration algorithm estimates M+N (=M+1+N-1) sensor poses. In addition to the map and the poses, the registration algorithm also estimates other variables like biases in sensor measurements, scale of the estimated poses, etc. The observability of variables to be estimated depends on the data collected for performing registration and also the measurement residuals used in the estimation process.

Given that a generic registration algorithm estimates sensor poses, map and extrinsic calibration between pairs of sensors, it can be stated that a generic registration algorithm is a multi-sensor calibration, localization and mapping algorithm. In theory the generic registration algorithm can be used both for estimation of the static extrinsic calibration parameters and also for localization & mapping. The quality of sensor calibration estimation depends on data provided to the registration

framework. In practice, before using the framework for localization & mapping it must be used to estimate the static calibration parameters and once an estimate of these parameters is obtained they can be used to initialize the localization & mapping process using the same framework. The generelizability of registration algorithms is demonstrated in Chapters 3, 4, 5 & 6 where in addition to static sensor registration, other valriables like evolving pose, biases, scale, etc. were also determined.

8.4.2 Bringing Part I & Part II together

Part I describes methods to perform multi-sensor registration of sensors mounted on a sensor suite that can be rigidly attached to a mobile platform, and Part II deals with registration of static sensors rigidly placed on infrastructure where autonomous vehicles operate. The mobile sensor suite described in Part I comprises commonly used sensors like LiDARs, IMUs and Cameras which are found in virtually all autonomous vehicles, ground robots, etc. Moreover, Part I primarily presents a filtering based technique to perform multi-sensor registration or more elaborately multi-sensor extrinsic calibration, localization & mapping using on board sensors, and Part II presents a batch-optimization based technique to achieve the same objective using both onboard and static infrastructure sensors. The filtering technques presented in Part I can be combined with the batch optimization and information theoritic techniques presented in Part II to bring together the advantages of fast estimation and global optimization, using both on-board sensing and off-board infrastructure sensing, in order to build autonomous systems that leverage sensor measurements from all available sensors with an aim to improve their perception and state estimation tasks that result in significantly positive downstream impacts like improved situational awareness, robust perception capabilities, immunity against sensor outage and increased public safety.

REFERENCES

[1] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "3d lidar-imu calibration based on upsampled preintegrated measurements for motion distortion correction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2149–2155, 2018.

[2] J. Zhang and S. Singh, "Loam : Lidar odometry and mapping in real-time," *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, 01 2014.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 15–22, 2014.

[4] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, October 2012.

[5] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[6] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *CVPR*, 2020.

[7] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2088–2095, 2012.

[8] F. M. Mirzaei and S. I. Roumeliotis, "A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1143–1156, 2008.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[10] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[11] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, p. 1330–1334, Nov. 2000.

[12] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3945–3950, 2007.

[13] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5695–5701, 2006.

[14] R. Bergelt, O. Khan, and W. Hardt, "Improving the intrinsic calibration of a velodyne lidar sensor," in *2017 IEEE SENSORS*, pp. 1–3, 2017.

[15] N. Muhammad and S. Lacroix, "Calibration of a rotating multi-beam lidar," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5648–5653, 2010.

[16] J. Lv, J. Xu, K. Hu, Y. Liu, and X. Zuo, "Targetless calibration of lidar-imu system based on continuous-time batch estimation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9968–9975, 2020.

[17] C. Glennie, "Calibration and kinematic analysis of the velodyne hdl-64e s2 lidar sensor," *Photogrammetric Engineering '&' Remote Sensing*, vol. 78, pp. 339–347, 04 2012.

[18] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor extrinsics and timing offset estimation," *IEEE Transactions on Robotics*, vol. 32, pp. 1215–1229, Oct 2016.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[20] J. Rehder, P. Beardsley, R. Siegwart, and P. Furgale, "Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 459–465, 2014.

[21] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laserrangefinder to a camera," 07 2005.

[22] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4164–4169, Oct 2007.

[23] L. Huang and M. Barth, "A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation," in *2009 IEEE Intelligent Vehicles Symposium*, pp. 117–122, June 2009.

[24] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3d laser scanner and an omnidirectional camera," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 336 – 341, 2010. 7th IFAC Symposium on Intelligent Autonomous Vehicles.

[25] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation," pp. 642–648, 06 2012.

[26] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," pp. 5562–5569, 10 2018.

[27] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, 09 2014.

[28] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," 06 2013.

[29] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[30] S. Garrido-Jurado, R. Muñoz Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, 10 2015.

[31] S. A. Rodriguez F., V. Fremont, and P. Bonnifait, "Extrinsic calibration between a multi-layer lidar and a camera," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 214–219, Aug 2008.

[32] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of rgb camera with velodyne lidar,"

[33] J. Kümmerle, T. Kühner, and M. Lauer, "Automatic calibration of multiple cameras and depth sensors with a spherical target," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, Oct 2018.

[34] J. L. Owens, P. R. Osteen, E. Corporation, and K. Daniilidis, "Msg-cal: Multi-sensor graph-based calibration."

[35] R. Ishikawa, T. Oishi, and K. Ikeuchi, "Lidar and camera calibration using motions estimated by sensor fusion odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7342–7349, 2018.

[36] S. Wasielewski and O. Strauss, "Calibration of a multi-sensor system laser rangefinder/camera," in *Proceedings of the Intelligent Vehicles '95. Symposium*, pp. 472–477, Sep. 1995.

[37] Qilong Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2301–2306 vol.3, Sep. 2004.

[38] R. Gomez-Ojeda, J. Briales, E. Fernandez-Moral, and J. Gonzalez-Jimenez, "Extrinsic calibration of a 2d laser-rangefinder and a camera based on scene corners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3611–3616, May 2015.

[39] O. Naroditsky, A. Patterson, and K. Daniilidis, "Automatic alignment of a camera with a line scan lidar system," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3429–3434, May 2011.

[40] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3936–3943, 05 2012.

[41] Q. V. Le and A. Y. Ng, "Joint calibration of multiple sensors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3651–3658, Oct 2009.

[42] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[43] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard, "CMRNet: Camera to LiDAR-map registration," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, oct 2019.

[44] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1110–1117, 2018.

[45] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," 2017.

[46] P. Jiang, P. Osteen, and S. Saripalli, "Semcal: Semantic lidar-camera calibration using neural mutual information estimator," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–7, 2021.

[47] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[48] R. Grompone von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 722–732, April 2010.

[49] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[50] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[51] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, May 2011.

[52] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[53] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "Lic-fusion: Lidar-inertial-camera odometry," *CoRR*, vol. abs/1909.04102, 2019.

[54] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[55] C. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *Journal of Machine Learning Research, v.11, 3011-3015 (2010)*, vol. 11, 11 2010.

[56] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration theory for fast and accurate visual-inertial navigation," *CoRR*, vol. abs/1512.02363, 2015.

[57] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4666–4672, 2020.

[58] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," vol. 3, pp. 2743 – 2748 vol.3, 11 2003.

[59] N. Trawny and S. Roumeliotis, "Indirect kalman filter for 3 d attitude estimation," 2005.

[60] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[61] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[62] M. Fleps, E. Mair, O. Ruepp, M. Suppa, and D. Burschka, "Optimization based imu camera calibration," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3297–3304, 2011.

[63] J. Kelly and G. S. Sukhatme, "Fast relative pose calibration for visual and inertial sensors," in *Experimental Robotics* (O. Khatib, V. Kumar, and G. J. Pappas, eds.), (Berlin, Heidelberg), pp. 515–524, Springer Berlin Heidelberg, 2009.

[64] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," 2020.

[65] X. Zuo, Y. Yang, P. Geneva, J. Lv, Y. Liu, G. Huang, and M. Pollefeys, "Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking," 2020.

[66] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

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

[68] Y.-J. Cho and K.-J. Yoon, "Distance-based camera network topology inference for person re-identification," *Pattern Recognition Letters*, vol. 125, pp. 220–227, 2019.

[69] R. Farrell and L. S. Davis, "Decentralized discovery of camera network topology," in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 1–10, IEEE, 2008.

[70] P. Chakravarty and R. Jarvis, "External cameras and a mobile robot: A collaborative surveillance system,"

[71] Z. Zou, R. Zhang, S. Shen, G. Pandey, P. Chakravarty, A. Parchami, and H. X. Liu, "Real-time full-stack traffic scene perception for autonomous driving with roadside cameras," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2022.

[72] K. Koide and E. Menegatti, "Non-overlapping rgb-d camera network calibration with monocular visual odometry," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9005–9011, 2020.

[73] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp.," in *Robotics: Science and Systems* (J. Trinkle, Y. Matsuoka, and J. A. Castellanos, eds.), The MIT Press, 2009.

[74] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

[75] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, vol. 19, p. 245–254, jul 1985.

[76] H. Chen, "A screw motion approach to uniqueness analysis of head-eye geometry," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 145–151, 1991.

[77] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Quaternion averaging," 2007.

[78] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, 2018.

[79] A. Vora, S. Agarwal, G. Pandey, and J. McBride, "Aerial imagery based lidar localization for autonomous vehicles," 2020.

[80] "Maxar technologies," 2020.

[81] A. G. Vora, S. Agarwal, J. N. Hoellerbauer, and F. Shaik, "High definition 3d mapping," June 6 2019. US Patent App. 15/831,295.

[82] R. W. Wolcott and R. Eustice, "Visual localization within lidar maps for automated urban driving," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–183, 2014.

[83] H. Yu, W. Zhen, W. Yang, J. Zhang, and S. Scherer, "Monocular camera localization in prior lidar maps with 2d-3d line correspondences," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4588–4594, 2020.

[84] A. Viswanathan, B. R. Pires, and D. Huber, "Vision based robot localization by ground to satellite matching in gps-denied situations," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 192–198, 2014.

[85] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[86] B. Li, L. Heng, K. Koser, and M. Pollefeys, "A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1301–1307, 2013.

[87] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[88] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[89] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," *CoRR*, vol. abs/1712.07629, 2017.

[90] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle, "Analysis and solutions of the three point perspective pose estimation problem," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 592–598, 1991.

[91] P. Viola and W. Wells, "Alignment by maximization of mutual information," vol. 24, pp. 16–23, 01 1995.

[92] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 187–198, 1997.

[93] Mathworks, “Roadrunner,” 2019.

[94] U. Engine, “Unreal Editor,” 2016.

[95] M. Li and A. I. Mourikis, “Online temporal calibration for camera–imu systems: Theory and algorithms,” *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 947–964, 2014.

APPENDIX A

CALCULATION OF JACOBIANS

A.1 Introduction

The perturbation model for the IMU pose is:

$$\tilde{R}_G^{I_k} = \texttt{Exp}(-\delta\theta_G^{I_k})\hat{R}_G^{I_k} \tag{A.1}$$

$$\tilde{p}_{I_k}^G = \hat{p}_{I_k}^G + \delta p_{I_k}^G \tag{A.2}$$

Here $R_{I_k}^G$ & $p_{I_k}^G$ are the IMU pose in a global reference frame. The perturbation model for the calibration parameters R_c & p_c are:

$$\tilde{R}_c = \texttt{Exp}(-\delta\theta_c)\hat{R}_c \tag{A.3}$$

$$\tilde{p}_c = \texttt{Exp}(-\delta\theta_c)p_c + \delta p_c \tag{A.4}$$

$\sim$ &$\hat{}$denote perturbed and nominal values respectively. $R \in SE(3)$, & $p, \delta p, \delta\theta \in R^{3\times 1}$ and $\texttt{Exp}(\delta\theta)$ maps a rotation vector $\delta\theta \in R^{3\times 1}$ on tangent space on to $SO(3)$.

A.2 Jacobians of Motion Based Calibration

The 6 DoF sensor (S's) motion between instants i & j is represented by a rotation matrix $R_{S_j}^{S_i} \in SE(3)$ and a translation vector $p_{S_j}^{S_i} \in R^{3\times 1}$.

$$R_{S_j}^{S_i} = R_c^\top R_G^{I_i} {R_G^{I_j}}^\top R_c \tag{A.5}$$

$$p_{S_j}^{S_i} = R_c^\top (R_G^{I_i}(p_{I_j}^G - p_{I_i}^G) + R_G^{I_i}{R_G^{I_j}}^\top p_c - p_c) \tag{A.6}$$

The Jacobians of the Rotation function with respect to the state variables are:

$$H_{\theta_c}^{R_{S_j}^{S_i}} = R_c{}^{\top}(R_G^{I_i} R_G^{I_j}{}^{\top} - I) \tag{A.7}$$

$$H_{\theta_{I_i}}^{R_{S_j}^{S_i}} = R_c{}^{\top} \tag{A.8}$$

$$H_{\theta_{I_j}}^{R_{S_j}^{S_i}} = -R_c{}^{\top} R_G^{I_i} R_G^{I_j}{}^{\top} \tag{A.9}$$

The Jacobians of the translation function with respect to the state variables are:

$$H_{\theta_c}^{p_{S_j}^{S_i}} = -R_c{}^{\top}[R_G^{I_i}(p_{I_i}^G - p_{I_j}^G) + R_G^{I_i} R_G^{I_j}{}^{\top} p_c - p_c]_{\times} + R_c{}^{\top}(R_G^{I_i} R_G^{I_j}{}^{\top} - I)[p_c]_{\times} \tag{A.10}$$

$$H_{p_c}^{p_{S_j}^{S_i}} = R_c{}^{\top}(R_G^{I_i} R_G^{I_j}{}^{\top} - I) \tag{A.11}$$

$$H_{\theta_{I_i}}^{p_{S_j}^{S_i}} = R_c{}^{\top}[R_G^{I_i}(p_{I_i}^G - p_{I_j}^G) + R_G^{I_i} R_G^{I_j}{}^{\top} p_c]_{\times} \tag{A.12}$$

$$H_{p_{I_i}^G}^{p_{S_j}^{S_i}} = -R_c{}^{\top} R_G^{I_i} \tag{A.13}$$

$$H_{\theta_{I_j}}^{p_{S_j}^{S_i}} = -R_c{}^{\top} R_G^{I_i} R_G^{I_j}{}^{T}[p_c]_{\times} \tag{A.14}$$

$$H_{p_{I_j}^G}^{p_{S_j}^{S_i}} = R_c{}^{\top} R_G^{I_i} \tag{A.15}$$

A.3 Jacobians of Reprojection Error Minimization based Calibration

The transformation of world coordinate points to camera coordinate frame is given as:

$$P_C = \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = R_W^C P_W + p_W^C \tag{A.16}$$

Where,

$$R_W^C = R_c^{\top} R_G^{I_k} R_G^{I_0}{}^{\top} R_c \tag{A.17}$$

$$p_W^C = R_c^{\top}(R_G^{I_k} R_G^{I_0}{}^{\top} - I)p_c + R_G^{I_k}(p_{I_0}^G - p_{I_k}^G) \tag{A.18}$$

The Jacobians of R_W^C & p_W^C w.r.t. state variables $T_{Ik}^G = \begin{bmatrix} {R_G^{I_k}}^\top & p_{I_k}^G \\ 0 & 1 \end{bmatrix}$ and $T_c = \begin{bmatrix} R_c & p_c \\ 0 & 1 \end{bmatrix}$ are:

$$H_{R_c}^{R_W^C} = {R_c}^\top ({R_W^{I_k}} {R_W^{I_0}}^\top - I) \tag{A.19}$$

$$H_{p_c}^{R_W^C} = 0 \tag{A.20}$$

$$H_{R_c}^{p_W^C} = -{R_c}^\top [{R_G^{I_k}} {R_G^{I_0}}^\top p_c + R_G^{I_k}(p_{I_0}^G - p_{I_k}^G) - p_c]_\times + {R_c}^\top ({R_G^{I_k}} {R_G^{I_0}}^\top - I)[p_c]_\times \tag{A.21}$$

$$H_{p_c}^{p_W^C} = {R_c}^\top ({R_G^{I_k}} {R_G^{I_0}}^\top - I) \tag{A.22}$$

$$H_{R_G^{I_k}}^{R_W^C} = {R_c}^\top \tag{A.23}$$

$$H_{p_{I_k}^G}^{R_W^C} = 0 \tag{A.24}$$

$$H_{R_G^{I_k}}^{p_W^C} = {R_c}^\top [R_G^{I_k}({R_G^{I_0}}^\top p_c + (p_{I_0}^G - p_{I_k}^G))]_\times \tag{A.25}$$

$$H_{p_{I_k}^G}^{p_W^C} = -{R_c}^\top R_G^{I_k} \tag{A.26}$$

The Jacobians of P_C w.r.t. state variables $T_{Ik}^G = \begin{bmatrix} {R_G^{I_k}}^\top & p_{I_k}^G \\ 0 & 1 \end{bmatrix}$ and $T_c = \begin{bmatrix} R_c & p_c \\ 0 & 1 \end{bmatrix}$ are:

$$H_{R_c}^{P_C} = [R_W^C P_W]_\times H_{R_c}^{R_W^C} + H_{R_c}^{p_W^C} \tag{A.27}$$

$$H_{p_c}^{P_C} = H_{p_c}^{p_W^C} \tag{A.28}$$

$$H_{R_G^{I_k}}^{P_C} = [R_W^C P_W]_\times H_{R_G^{I_k}}^{R_W^C} + H_{R_G^{I_k}}^{p_W^C} \tag{A.29}$$

$$H_{p_{I_k}^G}^{P_C} = H_{p_{I_k}^G}^{p_W^C} \tag{A.30}$$

The points in the camera frame are projected as:

$$z = \pi(P_C) = \begin{bmatrix} X_C/Z_C \\ Y_C/Z_C \end{bmatrix} \tag{A.31}$$

The Jacobian of π w.r.t. P_C is given as follows:

$$H^{\pi}_{P_C} = \begin{bmatrix} \frac{1}{Z_C} & 0 & -\frac{X_C}{Z_C^2} \\ 0 & \frac{1}{Z_C} & -\frac{Y_C}{Z_C^2} \end{bmatrix} \tag{A.32}$$

This leads to the following Jacobians:

$$H^{\pi}_{R_c} = H^{\pi}_{P_C} H^{P_C}_{R_c} \tag{A.33}$$

$$H^{\pi}_{p_c} = H^{\pi}_{P_C} H^{P_C}_{p_c} \tag{A.34}$$

$$H^{\pi}_{R^{I_k}_G} = H^{\pi}_{P_C} H^{P_C}_{R^{I_k}_G} \tag{A.35}$$

$$H^{\pi}_{p^G_{I_k}} = H^{\pi}_{P_C} H^{P_C}_{p^G_{I_k}} \tag{A.36}$$

A.4 Jacobians of Camera LiDAR measurement constraints

The measurement residuals are as given:

$$r_{CL} = \begin{bmatrix} r_R \\ r_p \end{bmatrix} = \begin{bmatrix} n_{C_j} - {R^I_C}^{\top} R^{Ij}_G {R^{Ii}_G}^{\top} R^I_L n_{L_i} \\ n^{\top}_{C_j} p^{C_j}_{L_i} + d_{C_j} - d_{L_i} \end{bmatrix} \tag{A.37}$$

Where $p^{C_j}_{L_i} = {R^I_C}^{\top} R^{I_j}_G ({R^{I_i}_G}^{\top} p^I_L + p^G_{I_i}) - {R^I_C}^{\top} (p^I_C + R^{I_j}_G p^G_{I_j})$. This residual constrains the detection of the calibration target at time i in the LiDAR sensor and the detection of the same calibration target at time j in the camera sensor.

The Jacobians of r_R w.r.t. state variables $T^G_{I_i} = \begin{bmatrix} {R^{I_i}_G}^{\top} & p^G_{I_i} \\ 0 & 1 \end{bmatrix}$, $T^G_{I_j} = \begin{bmatrix} {R^{I_j}_G}^{\top} & p^G_{I_j} \\ 0 & 1 \end{bmatrix}$, $T^I_L = \begin{bmatrix} R^I_L & p^I_L \\ 0 & 1 \end{bmatrix}$, and $T^I_C = \begin{bmatrix} R^I_C & p^I_C \\ 0 & 1 \end{bmatrix}$ are:

$$H^{r_R}_{R^{I_i}_G} = {R^I_C}^\top R^{I_j}_G {R^{I_i}_G}^\top [R^I_L n_L]_\times \tag{A.38}$$

$$H^{r_R}_{p^G_{I_i}} = 0 \tag{A.39}$$

$$H^{r_R}_{R^{I_j}_G} = -{R^I_C}^\top [R^{I_j}_G {R^{I_i}_G}^\top R^I_L n_L]_\times \tag{A.40}$$

$$H^{r_R}_{p^G_{I_j}} = 0 \tag{A.41}$$

$$H^{r_R}_{R^I_C} = {R^I_C}^\top [R^{I_j}_G {R^{I_i}_G}^\top R^I_L n_L]_\times \tag{A.42}$$

$$H^{r_R}_{p^I_C} = 0 \tag{A.43}$$

$$H^{r_R}_{R^I_L} = -R^I_C R^{I_j}_G {R^{I_i}_G}^\top [R^I_L n_L]_\times \tag{A.44}$$

$$H^{r_R}_{p^I_L} = 0 \tag{A.45}$$

Next, the Jacobians of r_p w.r.t. state variables $T^G_{I_i} = \begin{bmatrix} {R^{I_i}_G}^\top & p^G_{I_i} \\ 0 & 1 \end{bmatrix}$, $T^G_{I_j} = \begin{bmatrix} {R^{I_j}_G}^\top & p^G_{I_j} \\ 0 & 1 \end{bmatrix}$, $T^I_L = \begin{bmatrix} R^I_L & p^I_L \\ 0 & 1 \end{bmatrix}$, and $T^I_C = \begin{bmatrix} R^I_C & p^I_C \\ 0 & 1 \end{bmatrix}$ should be determined, but before that, the Jacobians of

$p_{L_i}^{C_j}$ w.r.t. the above mentioned state variables need to be determined. The Jacobians of $p_{L_i}^{C_j}$ are:

$$H_{R_G^{I_i}}^{p_{L_i}^{C_j}} = -{R_C^I}^\top R_G^{I_j} {R_G^{I_i}}^\top [p_L^I]_\times \tag{A.46}$$

$$H_{p_{I_i}^G}^{p_{L_i}^{C_j}} = I \tag{A.47}$$

$$H_{R_G^{I_j}}^{p_{L_i}^{C_j}} = {R_C^I}^\top [R_G^{I_j}({R_G^{I_i}}^\top p_L^I + p_{I_i}^G - p_{I_j}^G)]_\times \tag{A.48}$$

$$H_{p_{I_j}^G}^{p_{L_i}^{C_j}} = I \tag{A.49}$$

$$H_{R_C^I}^{p_{L_i}^{C_j}} = -{R_C^I}^\top [R_G^{I_j}({R_G^{I_i}}^\top p_L^I + p_{I_i}^G - p_{I_j}^G)]_\times \tag{A.50}$$

$$H_{p_C^I}^{p_{L_i}^{C_j}} = -{R_C^I}^\top \tag{A.51}$$

$$H_{R_L^I}^{p_{L_i}^{C_j}} = {R_C^I}^\top R_G^{I_j} {R_G^{I_i}}^\top [p_L^I]_\times \tag{A.52}$$

$$H_{p_L^I}^{p_{L_i}^{C_j}} = {R_C^I}^\top R_G^{I_j} {R_G^{I_i}}^\top \tag{A.53}$$

The Jacobians of $p_{L_i}^{C_j}$ will be useful in determining the Jacobians of r_p w.r.t. state variables $T_{I_i}^G = \begin{bmatrix} {R_G^{I_i}}^\top & p_{I_i}^G \\ 0 & 1 \end{bmatrix}$, $T_{I_j}^G = \begin{bmatrix} {R_G^{I_j}}^\top & p_{I_j}^G \\ 0 & 1 \end{bmatrix}$, $T_L^I = \begin{bmatrix} R_L^I & p_L^I \\ 0 & 1 \end{bmatrix}$, and $T_C^I = \begin{bmatrix} R_C^I & p_C^I \\ 0 & 1 \end{bmatrix}$ as follows:

$$H^{r_p}_{R^{I_i}_G} = n_c^\top H^{p^{C_j}_{L_i}}_{R^{I_i}_G} \tag{A.54}$$

$$H^{r_p}_{p^G_{I_i}} = n_c^\top H^{p^{C_j}_{L_i}}_{p^G_{I_i}} \tag{A.55}$$

$$H^{r_p}_{R^{I_j}_G} = n_c^\top H^{p^{C_j}_{L_i}}_{R^{I_j}_G} \tag{A.56}$$

$$H^{r_p}_{p^G_{I_j}} = n_c^\top H^{p^{C_j}_{L_i}}_{p^G_{I_j}} \tag{A.57}$$

$$H^{r_p}_{R^I_C} = n_c^\top H^{p^{C_j}_{L_i}}_{R^I_C} \tag{A.58}$$

$$H^{r_p}_{p^I_C} = n_c^\top H^{p^{C_j}_{L_i}}_{p^I_C} \tag{A.59}$$

$$H^{r_p}_{R^I_L} = n_c^\top H^{p^{C_j}_{L_i}}_{R^I_L} \tag{A.60}$$

$$H^{r_p}_{p^I_L} = n_c^\top H^{p^{C_j}_{L_i}}_{p^I_L} \tag{A.61}$$