Simultaneous Localization Auto-Calibration and Mapping of Ground Vehicles

by

Jordan Britt

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
Aug 6, 2016

Keywords: SLACAM, SLAM, Auto-Calibration, Lidar

Approved by

David M. Bevly, Chair, Professor of Mechanical Engineering
Dan Marghitu, Professor of Mechanical Engineering
Subhash Sinha, Professor of Mechanical Engineering
George Flowers, Professor of Mechanical Engineering and Dean of Graduate School

Abstract

This dissertation advances the state of the art of sensor auto-calibration by presenting a generic solution and analysis to the 3D sensor auto-calibration problem, as well as to the 2D simultaneous localization and auto-calibration (SLACAM) problem . Proper sensor calibration can be key to mission success when attempting to navigate robots in challenging environments. It allows for maximum information correlation between two sensors leading to higher fidelity representations of the environment thereby facilitating more precise navigation and obstacle avoidance, without which sub-optimal solutions such as taking longer paths to avoid obstacles to account for sensor error and other similar work arounds are employed. Additionally the ability to calibrate on the fly can greatly increase the robustness of a robot operating in challenging environments where canceling a mission prematurely is simply infeasible or not an option.

The ability to auto-calibrate two rigidly mounted sensors while on a moving platform will be validated using a rigidly mounted lidar and an inertial navigation system (INS) on a sport utility vehicle (SUV) while driving through an urban center. The presented technique will be compared to an independent party's static laboratory calibration, which will show agreement to within tenths of degrees in angular accuracy and centimeter level translational accuracies. Additionally, the observability of this technique will be assessed and unobservable maneuvers will be highlighted. Additionally the three (DOF) alignment of an inertial measurement unit (IMU), lidar, and odometer system on a ground robot that is performing simultaneous localization and mapping (SLAM) will be assessed. Additionally the performance of the SLAM algorithm will be assessed as well as the observability of this system. This technique will demonstrate the ability to produce a navigation solution that accurately navigates the ground robot through a structured environment to within a robots

length of the initial position while calibrating the lidar and odometer systems relative to the IMU to within the state of the art of single sensor calibration techniques.

Acknowledgments

I would like to begin by thanking both my committee and my advisor Dr. Bevly specifically. I truly appreciate both the time and effort that was put in to making this dissertation better, the guidance provided, and encouragement. If not for the thought provoking insight provided both in and out of the classroom, my graduate experience would have been surely poorer. I would also like to acknowledge the impact that my fellow lab mates. It was truly fantastic to be surround by such a large group of intelligent, patient people who were constantly willing to stop what they were doing to help you regardless of how late it was or how pressing the current deadline. Additionally, it was these people that helped make the experience fun. I learned so much from being surround by these people who asked questions I hand't yet thought of and who introduced new ideas and ways of doing things that I was unfamiliar.

Finally I would like to acknowledge and thank my family, especially my wife. You have all been an amazing and continued source of encouragement and support, and I am truly blessed to have had you as part of my life.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

EKF extended Kalman filter

FOV Field of View

ICP Iterative Closest Point

IMU Inertial Measurement Unit

lidar Light Detection and Ranging

PF Particle Filter

SAM Split and Merge

SLACAM Simultaneous Localization, Auto-Calibration, and Mapping

SLAT Simultaneous Localization and Tracking

TLS Total Least Squares

UUV Unmanned Underwater Vehicle

ZUPT Zero-Velocity Update

Chapter 1

Introduction

1.1 Motivation

Modern navigation systems typically rely on the output of multiple sensors to obtain a navigation solution. Typically these sensors are not rigidly mounted on top of one another and are often distributed throughout the mobile platform with the goal of providing an increased perception footprint thereby facilitating greater situational awareness. However, these sensors inherently only measure in their individual sensor frames which can be specific to the sensor in both the coordinate frame used, the direction of measurement, and measurement type. Therefore in order to accurately represent the environment that these sensors are measuring, the output of these sensors needs to be transformed into a single common coordinate frame. To accomplish this, it is necessary to know both the translation and rotation between the desired common coordinate frame and the coordinate frame of the sensor. Once this information is known, the sensor can then be transformed into the desired common coordinate frame.

Unfortunately obtaining this alignment is not a trivial task. Ideally one would simply measure the offsets in the mounting location and rely on the drawings from the manufacturer to divine the mounting locations. This approach neglects the difficulty in obtaining precise measurements through barriers on non planar surfaces as well as any manufacturing errors. Additionally there are some sensors such as those from skid-steered odometry that simple measurements of wheelbase paint an incomplete picture of the true dynamics that the robot will experience. Therefore, these simple measurement techniques are coupled with calibration targets to offset errors obtained in the hand measurement process [53, 97]. While this approach can obtain high fidelity calibrations, it is often arduous, requiring both significant

time, cost, equipment, and technical expertise. Furthermore this calibration procedure due to its large initial investment in time and effort is assumed to remain constant. This is inherently a flawed assumption . Typically in the course of a robot's life, sensors are reattached, parts wear - such as tires, and the mass of the robot is altered, all events which can alter the calibration parameters of sensors [16]. Additionally robots often operate in challenging environments which may not be safe for humans, and during the course of a mission it is not uncommon for sensors to be jarred as they navigate the environment, thereby altering the sensor calibration. However ending a mission prematurely can be infeasible or impossible and therefore the ability to calibrate a sensor on-the-fly without the aid of human intervention is a necessary means of increasing overall system robustness. As such, some form of auto-calibration has been employed for interplanetary exploration [35], for the exploration of volcanoes [14], and has been sought by the military [57] to increase system robustness while unburdening the war fighter of any needed technical expertise of sensor calibration.

However, auto-calibration schemes are often developed to be very specific not only to the sensor types to be calibrated, but also specific to a given sensor. Hence auto-calibration that works for lidar A may not work for lidar B. This is illustrated in [97] where a camera and lidar system are being calibrated by exploiting the fact that this specific lidar operates at a frequency that the camera is capable of seeing, and would not be possible if substituted for an alternative camera system or lidar. This lack of generality imposes an unnecessary barrier to auto-calibration, requiring different algorithms to calibrate a platform thereby diminishing the ease at which sensor can be exchanged or upgraded.

Often, the sensors on a platform are used for navigation in addition to merely mapping or environmental perception alone. One common application is the simultaneous localization and mapping (SLAM) problem in which a robot has no a priori information about its surroundings and will seek to create a map of its environment while simultaneously navigating itself through the environment with the aid of the map it has just created [22, 21]. General solutions to this problem seek to map obstacles from the sensor frame into the navigation

frame in an effort to construct a map, while keeping track of measurement errors which is critical to the success of SLAM. Therefore having an unknown alignment between the robot and the sensor compounds the SLAM problem and the error in both the environmental measurement as well as the estimated mounting error must be carefully discerned from one another and dealt with appropriately. The process of performing auto-calibration along side SLAM is known as simultaneous localization, auto-calibration, and mapping (SLACAM) . Auto-Calibrating while on a moving platform that is operating in an unknown environment creates a compounding problem due to most auto-calibration schemes requiring accurate historical information, and SLAM often requiring accurate environmental perception. This is often overcome by calibration schemes assuming a minimal amount of error such as only a degree and a few centimeters of initial error. This however lacks the robustness desired of the ideal SLACAM scheme where the sensor would be capable of being calibrated in any unknown position and the robot still capable of solving the SLAM problem. Additionally, most research that focuses on the SLACAM problem tends to focus on the sheer ability to perform the auto-calibration step while not discussing the impact this has on SLAM performance, but merely defining success as non-divergence of the SLAM algorithm. Therefore a more rigorous standard must be met for practical operation.

1.2 Contributions

While most auto-calibration techniques are specific to a subgroup of sensors such as lidar camera calibration or INS GPS calibration, the technique presented in this dissertation is unique in that it makes no inherent assumption as to the types of sensors to be calibrated, their raw measurements, and relies minimally on historical information. However, this technique does assume that each sensor is capable of deriving its change in pose (position and orientation) from successive measurement epochs. Therefore while neither lidar nor wheel-encoders make raw measurements of their respective change in pose, they still however can be calibrated to one another via the use of iterative closest point (ICP) for

the case of the lidar, and a kinematic approach for the wheel encoders. This technique is therefore quite generic allowing it to be applied to a number of different sensor calibration problems. However, because of this approach, sensors that are prone to erroneous measurements will inherently provide poorer calibration results than sensors of higher quality. Using each sensor's relative pose estimate, an optimal translation and rotation can be found such that the perceived motion of one sensor matches the other. It is this rotation and translation that will constitute the desired calibration parameters. Because this technique requires that each sensor measure its relative pose, it is well suited for dynamic platforms and calibration on-the-fly.

This technique will be analyzed in both the 6-DOF and 3-DOF cases where for the 6-DOF frame a 3D-lidar will be calibrated to an INS system. These sensors will be mounted on an SUV and driven around an urban center so that the performance of this algorithm can be assessed in a real-world scenario. Both simulated and actual calibration results for this test will be presented as well as the various observable and unobservable trajectories of this system. The final calibration results will be compared to an independent laboratory calibration of the sensor suite. In the 3-DOF case, the developed techniques will be used to calibrate a lidar, an IMU, and wheel odometry to a mobile robot while the mobile robot performs SLAM. While a similar truth method to that of the 6-DOF cases is lacking, the performance of the SLACAM system will be assessed on the ability of the robot to successfully return to the same location as it started in its map. The sensitivity to an initial error for each calibration parameters will be assessed to determine the combined error tolerance of the SLACAM system.

Therefore, this his work distinguishes itself from previous research by:

- Simulated and experimental validation of a generic auto-calibration technique operating in real-world conditions over an extended test period.
- Confirmation of experimental auto-calibration results through the use of an independent laboratory calibration.

- Comparison of Calibration system to other known and accepted techniques
- Performing SLACAM on the entire sensor navigation suite as opposed to merely some combination of the available sensors.
- Observability analysis of the SLACAM strategy provided this auto-calibration technique is used.
- Analysis of of SLACAM performance as function of initial sensor error in both physical reading as well as from which sensor an error can be tolerated the best.

1.3 Dissertation Organization

This dissertation is organized as follows. In Chapter 2, relevant background information on the various facets of SLACAM is given to provide a more thorough and nuanced understanding of various sensor to sensor calibrations, SLAM strategies, and how the two are merged to form SLACAM. This is followed by the development of the 6 DOF auto-calibration strategy in Chapter 3. Additionally, this chapter outlines how each of the individual sensors fits into this auto-calibration strategy. Simulated and experimental results for the 6 DOF strategy are then presented in Chapter 4 for this auto-calibration strategy. This is followed by the incorporation of auto-calibration into SLAM to outline the general SLACAM architecture and observability analysis in Chapter 5. In Chapter 6 simulated and experimental results are provided for the SLACAM strategy, and finally Chapter 7 provides final thoughts, conclusions, and future work.

Chapter 2

Background

This chapter provides an overview of the current state of the art for the SLACAM field of research. In order to accomplish this, a brief overview of the operation of the future sensors to be calibrated will be provided Section 2.1 in an effort to relay how these techniques relate practically to the sensor obtaining the measurement. Additionally an overview of various sensor to sensor calibration techniques will be provided in Section 2.2 so that a fluid progression from auto-calibration to SLACAM can be understood. Naturally an overview of SLAM will then be provided in Section 2.3, followed finally by an overview of SLACAM and its various implementations in Section 2.4.

2.1 Sensor Overview

While a host of different sensors are used are to navigate robots indoors, the three sensors that constitute the core of ground navigation, especially indoor, are the inertial measurement unit, wheel odometry, and vision. Vision sensors typically encompass both lidar and camera sensor, latter of which will the focus of this dissertation. An overview of the how these sensor are used for navigation and the raw measurements that they obtain will be provided, as well as the typical parameters that are sought for these sensors that constitute a "calibrated" sensor.

2.1.1 Inertial Navigation Background

An inertial measurement unit (IMU) is a self-contained device that is capable of measuring specific force and rotation rates and often defines the core of a robot navigation strategy. These specific forces are measured by accelerometers, also known as accels, can

then be integrated with respect to time to obtain velocity and integrated again to obtain the change in position for the integration period. Similarly the rotation measurements provided by gyroscopes, also known as gyros, can be integrated with respect to time to obtain the change in orientation over the integration period. It is important to note that these rotation and translational measurements are only provided in the local IMU frame, and simply denote how the IMU changes pose from measurement epoch to measurement epoch. Typically IMUs provided measurements at a high rate - often double to an order of magnitude faster than other sensors on the robot. While IMUs vary wildly in performance, typical robotic navigation is performed with tactical grade or MEMS grade IMUs. Tactical grade IMUs are capable of being used without the aid of additional sensors for several minutes, while MEMS grade IMUs can only provided accurate information without additional aid for several seconds. Although these sensors have inherent errors such as white noise on sensor measurements and moving biases, typically auto-calibration seeks to find the relative pose between this sensor and the sensor to be calibrated.

2.1.2 Odometry based Navigation and Calibration Background

Odometry in the context of this dissertation refers to the wheels or tracks of a vehicle that are in contact with the ground which the robot uses to propel itself as well as change directions. The auto-calibration work in this dissertation is focused on skid-steer, however these principles can be adapted to Ackerman steer or differential drive vehicles. Skid-steer is a locomotion strategy in which wheels or tracks are aligned on opposite sides of the vehicle. The robot can be steered by driving one set of wheels at a different rate than its opposing side. Generally there is an encoder on the driveshaft of the motor or axle from which can be used to determine the rotational change of the shaft to which it is connected. To ascertain the motion of the robot, typically the effective radius of the wheels needs to be determined as well as the geometry of the wheels, specifically the distance between wheels. Additionally, because robots are typically not perfectly symmetrical due to either mounting errors or offsets

in weights, the point at which the robot rotates about is desired both side to side as well as front to back. Additionally, when speaking of auto-calibration of odometry in reference to another sensor such as an IMU, the aforementioned parameters of wheel radii and geometry are desired as well as the location at which the robot moves about due to odometry relative to the other sensor. Note that this dissertation is focused on skid steered locomotion and calibration. It is worth noting that experiments such as [77] have shown the the proper blending of odometry and additional sensor information has shown great improvement over what either single sensor system could achieve individually.

A high fidelity model of skid steering for a tracked vehicle is developed in [93]. This work assumes the vehicle is operating on firm ground in steady state and takes into account the general kinematic description of instantaneous centers of rotation (ICR), the steer-shear displacement relationship on the track-ground interface, the variation of sprocket torques for the inner and outer tracks, and track pressure distribution. This model is validated against real data and shows excellent agreement, clearly demonstrating the importance of modeling skid-steer robots using ICR.

2.1.3 Lidar based Navigation and Calibration Background

Light detection and ranging (LIDAR) is a measurement technique based off of echo-location. Essentially, a pulse of light is transmitted and based on the time between transmission and the echo of that pulse the distance between the receiver of the lidar and the obstacle that created the reflecting pulse can be determined. Lidars are capable of making measurements in all three dimensions. A 1-D lidar is simply what was just described of a laser sensor making a single distance measurement. For increased perception, the laser beam of the lidar can be directed with a mirror that rotates with the aid of an electric motor. Using an encoder, the angle that the lidar is measuring as well as the distance to an obstacle can be ascertained. Finally, most 3-D lidars incorporate multiple 1-D lidars aligned in a plane which makes measures in a classical fan shape. This plane of lidars is then rotated about a

common axis so that measurements are obtained in all three directions, typically reported in polar coordinates. Additionally, most lidars are also capable of measuring the reflectivity of the surface measured. This measurement is commonly known as "echo width".

Note however that the sole function of the lidar is to make distance and reflectivity measurements which provides no inherent navigation information. In order to navigate using a laser scanner, the point clouds produced by the laser scanner at two different points in time must be compared so that motion can be inferred. The defacto standard is iterative closest point (ICP) which was initially introduced in [17] and [4]. In this technique the raw laser data from one scan is treated as a model for the environment. A separate scan measures roughly the same environment, where every point in mapped to a point in the model scan. The optimal rotation and translation is then found by minimizing the distance between these points, often in an iterative fashion such that the correspondences between points is assessed at each iteration. This method has many variants and has seen a wealth of research. Notable improvements have come in the form of k-d trees, a data-structure for representing the point-clouds, which has allowed for significant processing improvements was shown in [80]. The incorporation of lidar reflectivity to increase robustness as seen in [78]. Additionally some techniques are even cable of handling moving objects it the environment, where they are either removed from the environment, or tracked themselves [25].

Additional techniques have been developed in an attempt to speed up the sometimes arduous process of point matching through the use of features. Features are typically geometric descriptors of the environment such as lines [7], arcs [33], or planes [91]. These feature descriptors are extracted from the raw lidar data, such that points which closely resemble the desired features are saved. These saved features from scan to scan are then compared to one another to assess robot movement. Such techniques have been applied with success primarily in indoor navigation. Additional techniques such as those presented in [1] seek to

represent the environment using splines, and thus incorporate potentially more of the environment without the rigidity of the line or plane representation while having less overhead than the pure point cloud approach.

While lidars do have inherent measurement errors in both range and direction, these errors are intrinsic and not solved for by the inherently extrinsic auto-calibration scheme which is the focus of this dissertation. Because lidars are inherently self contained sensors from which navigation information in processed, extrinsic auto-calibration for lidars is focused on the relative pose in relation to another sensor or point on a vehicle.

2.2 Multi-Sensor Auto-calibration Background

This section will provide an overview of sensor to sensor calibration techniques specific to the sensors addressed in this dissertation. Specifically, IMU to odometry, and lidar to odometry calibration will be addressed as well as lidar to IMU calibration. These basic sensor to sensor techniques are generally those that are sought to be incorporated into the larger SLACAM framework, thus requiring further investigation. Note that some techniques refer to a vision system instead of a lidar specifically. In these instances, the specific calibration technique may have used a stereo-camera or similar device, however the concept is interchangeable.

2.2.1 IMU to Odometry

Due to the nature of both odometry and IMU sensors having errors that compound with time, few attempt to jointly calibrate these two sensors. Instead one is more typically substituted for another when calibration is concerned, or the alignment between the two sensors is assumed to be known.

In [2] the desired odometry parameters are found with the aid of an IMU. Their strategy is to effectively integrate the IMU for very short periods of time and compare these results to the odometry model. The odometry is modeled simply as a matrix of random variables

where no physical explanation is attempted to link these variables to natural phenomena. Least squares is used to find an optimal agreement between the odometry and IMU sensors. Truth was a trailering test, similar to that introduced in [6].

A generic calibration technique is introduced in [12] that allows two sensors capable of reporting relative change in pose to be calibrated. While not specifically demonstrated with IMU to odometry calibration, the author does hint at this possibility.

2.2.2 Lidar to Odometry

The University of Michigan benchmark (UMBmark) test is a landmark odometry calibration technique introduced in [6]. The UMBmark test provides a calibration strategy for resolving systematic odometry errors in a differential driver robot, specifically the wheel radius error and the wheelbase error. The test consists of driving the robot in a four meter by four meter square five times in both the clock-wise and counter-clock-wise directions with zero point turns occurring at the vertices. Errors are then separated into errors that affect both directions, and errors that affect only one direction, which are analogous to rotational and translational errors. Note that in this work sonar sensors are used to ascertain the true amount of rotation in the turns, however this could have also easily been accomplished with lidar. The calibration values are then calculated such that the robot returns to its starting location. This work shows o10-fold improvements over initial calibration estimates.

An odometry calibration scheme is introduced in [51] where the desired parameters are found using a genetic algorithm, by comparing the true robot path generated via lidar and GPS to the path reported by odometry. The specific calibration parameters sought are the instantaneous centers of rotation in both the forward direction and the lateral direction for each wheel as well as scale factor terms for the wheel radii. It is also noted that if the operator assumes that the left and right centers of rotation are the same, then the user is assuming that no slippage is occurring which can be confirmed by a steering efficiently index. This kinematic approach was validated on hard ground outdoors driving at moderate speeds.

Additionally, plots were shown to validate the assumption that the instantaneous centers of rotation stay in a bounded area confirming previous kinematic assumptions.

An odometry calibration scheme is analyzed in [46] which builds on [51] where first a model of a skid-steer robot is introduced. This model accounts for slip by allowing for the robot to operate about bounded instantaneous centers of rotation and the wheel radius to vary based on a scale factor. The optimal calibration parameters are found by having the robot drive a known path recorded by an overhead camera, and then these scale factors and instantaneous centers of rotations are found via a genetic algorithm. This method is validated by having the robot drive outdoors on varying surfaces with different tires. The true path as obtained by differential GPS is then compared with the calibrated values obtained via the genetic algorithm. Note that [46] also builds on previous work where this kinematic model is simply modeled as a matrix of unknowns which was solved using a least-squares and a truthing system found in [2].

Another auto-calibration scheme for odometry to a camera is introduced in [47] where the extrinsic parameters of the camera are sought. A point light source is defined as the origin and is tracked to determine the angular change of the robot. This angular change is compared with the odometry information to determine the desired parameters. The observability of this technique is also analyzed and is found to be observable. Note the common thread throughout these works, which generally relies on the lidar to act as a truthing system and record the true path of the robot, from which the odometry is calibration parameters are found, typically offline.

2.2.3 Vision to IMU

Often lidars are statically calibrated where special calibration targets are used that can be easily located within the reference frame of the vehicle, where it is assumed that the IMU is perfectly mounted. Therefore, when the lidar scans the calibration target, the desired pose parameters can be determined. Such a technique is presented in [3] where specially

constructed polypods are used to calibrate a 2D lidar in a laboratory setting. However, generally for vision to IMU calibration, the vehicle undergoes some dynamic maneuver.

A similar technique to [64] is introduced in [45] based on Rényi Quadratic Entropy. This technique is shown to calibrate both a 3D lidar to an INS system by tracking the relative pose of a vehicle and the corresponding point clouds at those poses. The optimal mounting parameters are defined as those that maximize the "crispness" of the map of the lidar scans. Additionally this technique is shown to apply to lidar-to-lidar calibration as well. Results were analyzed by comparison to hand measurements as well as a comparison of the variance of the mounting parameters found. Note that this technique assumes that the environment is static and that best results are obtained when all axes are accelerated.

The calibration strategy presented in [42] attempts to calibrate a 3D laser scanner to a vehicle's inertial frame. The technique makes the assumption that due to the density of the lidar scans, most surfaces will appear locally planar. Therefore, the algorithm penalizes points that do not meet this assumption. Point clouds are saved along with relative vehicle poses, and the optimal mounting parameters are defined as the ones that provide the most planar representation of the environment. The success of this algorithm was determined by both multiple initialization tests and physically moving the sensor where the initial error in the mounting estimate was on the order of 10cm and 1°.

Other lidar techniques require scanning the same known area several different times from varying perspectives. Such lidar calibration was performed in [69] where a lidar mounted in an aircraft was flown over the same area but using different angles of approach. The shift in the created point cloud from one direction to another allowed operators to determine the necessary rotational correction to the lidar that allowed the scans to align as expected given the path data provided by the IMU. This specific type of calibration, is known as boresight alignment.

A lidar calibration technique is presented in [87] where the lidar is calibrated with the aid of a pole to correct for decimeter level errors. Essentially, a pole covered in retro-reflective

tape is placed perpendicular to a flat segment of ground. The ground plane and pole are then segmented from one another. The pole is viewed from several different perspectives, and the algorithm then seeks the values that provide the best estimation to the given scene. The INS is assumed to be aligned with the body frame, and thus determining the correct angular mounting of the lidar, will align it with the inertial frame. While this algorithm does require the vehicle to perform dynamic maneuvers, it is noted that during the yaw calibration process, the vehicle is assumed to undergo no pitch or roll. However some non zero roll or pitch must occur for the full calibration to be successful.

An extrinsic calibration technique is introduced in [98] where the geometry of the environment is analyzed to determine the mounting of the lidar on a vehicle. Specifically the assumption is made that the vehicle is operating on a planar environment. Therefore, the ground plane is extracted from the lidar scan and the pitch, roll, and height of the lidar relative to this ground plane is deduced. The vehicle is then driven and pole like obstacles are extracted from the environment. Based on the motion of the pole like obstacles, the yaw and translational offsets can be calculated. The performance of this method was verified by hand measurements.

An extremely thorough introduction of manifolds and their incorporation with standard state estimation techniques is presented in [29]. This technique shows how manifolds can be incorporated into both least squares estimation as well as the unscented Kalman filter. [29] builds on [88] where a number of different sensors are calibrated through the use of Manifolds, including both a camera to robotic arm and a Microsoft Kinect to an accelerometer. These calibration parameters are found through effectively modeling the calibration parameters as a manifold, and solving this manifold as a nonlinear least-squares problem. Additionally a framework to solve these problems is presented known as the Manifold Toolkit for Matlab (MTKM) so that one simply needs to add relative motion estimates to the toolbox to model and solve the manifold problem.

2.3 SLAM Background

The simultaneous localization and mapping or SLAM problem of a robot building a map of its environment and accurately locating itself within and expanding that map through exploration without the aid of human intervention has been viewed as a "holy grail" of navigation for the robotics community prior to the late 1980's [21]. Prior to this, the two problems of localization and mapping where investigated independently. However the breakthrough papers [82, 22, 81] established the statistical cross-correlation that must be maintained between landmarks that allowed for the successful implementation of SLAM. It is from these few initial works and the coining of the term in [9], that has spawned waves of research in the robotics community. While research into the SLAM problem is vast in terms of both breadth and depth, two of the major areas of research have been been feature detection / extraction and filter structure. The sub-sections below should familiarize the reader with some of the more common filter structures used within the SLAM community as well as general concepts of detecting and using landmarks.

2.3.1 General Filter Structure Review

When investigating the various implementations of SLAM, it is important to recall the major goal of SLAM is to relate the various landmark locations to both the current position of the robot as well as the locations of other landmarks. Therefore, these various filter structures seek to accomplish this goal while simultaneously increasing either ease of implementation, speed, or robustness. Because SLAM inherently must keep track of all vehicle states as well as all observed landmarks, these are non-trivial considerations.

2.3.1.1 EKF SLAM

Extended Kalman filter (EKF) SLAM formulates the problem entirely around the Extended Kalman filter framework. That is to say that the time update is performed for both the robot position and landmark locations via the integration of kinematic equations. Then

the measurement update for both robot and landmark locations are computed via a linearization of the non-linear measurement equations about the current perceived operating point. This method is the root of most SLAM implementations, and has shown to be successful, often the benchmark for other SLAM implementations [86]. Despite its success, and ease of implementation, EKF SLAM is not without downsides. Because each time-step requires all states and covariances be updated with each measurement, the required computation time can grow significantly as landmarks are added. The EKF SLAM implementation is also quite sensitive to incorrect data association and errors in heading, which if large can also cause filter divergence. [58, 21].

Issues faced in EKF SLAM are outlined in [40], where it is noted that practically, EKF SLAM will always face some amount of inconsistency when compared to the idealized case simply because eventually, sensor errors will cause a false matching of landmarks. This inconsistency problem is explored through the use of observability analysis, where it is noted that while lines are often more robust than point features for navigation, they offer no inherent improvement in consistency. Additionally, it was noted that EKF SLAM is robust in that it can function in many different environments provided that two visible features can be maintained.

Additionally EKF SLAM is tested in an outdoor environment with spurious features using an offline dataset available with ground truth in [66]. EKF SLAM is demonstrated to function even in these challenging environments, and several possibilities are outlined for increased speed, such as using local maps, and the ability to stitch these together to create a global map in linear time.

2.3.1.2 Graph Based SLAM

Graph SLAM essentially seeks to solve the SLAM problem by representing all observed information as a graph. Essentially all measurement locations are treated as nodes, where various nodes are linked together through control inputs, representing change in motion.

Additionally each node links to any landmarks observed while the robot was at that node. Another way to envision this, is by treating nodes and measurements as masses connected via springs, where the optimal solution seeks to nudge the masses into optimal alignment. Essentially the observation and change in motion measurements are treated as constraints which are solved typically offline, through least squares, or similar techniques.

An application of graph SLAM is introduced in [62]. In [62] the relative motion at each time-step is logged and a graph/grid is constructed that logs these various changes in motion. Effectively a non-linear least squares is then applied to attempt to find the optimal path taken as well as to perform loop closure. This method was tested in both simulation and with real data and was shown to be as effective as EKF based SLAM in the real world dataset.

Similarly Graph SLAM was tested using several large outdoor environments in [86]. It was noted that when only the path of the robot was desired and not the map, computation time was quite fast, often less than the time it took to gather the data. However, reconstruction of the environment was also shown to be successful using this technique, albeit with the obvious drawback of being limited to post-processing.

2.3.1.3 Fast SLAM

Fast SLAM is a merger of EKF SLAM with a particle filter (PF) in an effort to increase the computational efficiency of SLAM. Essentially, a particle filter is used to update the robot position. Based on this updated position, a single EKF is implemented for each landmark update. By negating the cross-correlations of the landmark update, computational speed is significantly increased. This general approach is also referred to as a Rao-Balackwellized filter.

Such a technique is employed in [54] where a ground robot travels a course used to simulate exploration on Mars. This technique is shown to be capable of retaining thousands of landmarks while providing a SLAM solution. While data was presented in graphical form,

it was noted that there are diminishing returns when tracking large number of landmarks, and few improvements where made to robot position once one-hundred landmarks had been located.

This is similarly demonstrated in [75] where a small differential drive robot with a laser scanner employes fast SLAM to navigate the robot in an indoor environment. Using approximately 50 particles, the robot is able to close the loop successfully in a 300 m path. The path is merely compared with a map which shows good agreement. Additionally the approach was confirmed in simulation.

2.3.2 Landmark Detection

Because SLAM relies so heavily on the accurate representation of the environment and obstacles, a sensor capable of acquiring significant perception information such as a lidar or camera system is often employed. While SLAM has been shown to be theoretically possible with the use of less information rich sensors such as sonar [39], the use of such sensors as the primary environmental sensor is not the norm. Generally when deciding on what type of landmark to attempt to detect, it is often necessary to have an a priori expectation about the type of environment the robot will be operating. When the operating environment is totally unknown the point clouds themselves can be used, however when operating indoors, it is more typical to use straight line or curved features to aid navigation.

Occasionally, multiple detection techniques can be applied in SLAM. Such a technique leveraging redundant sensor information between a lidar and camera is presented in [15]. It was shown that when redundant sensor information is used, the overall performance of SLAM is improved not only in fewer false data associations but also fewer erroneous landmarks being tracked, thereby decreasing computational overhead. Additionally the author remarked on the need for quality sensor to sensor calibration for this algorithm to function properly. Sensor calibration for the work of [15] was performed statically using a black and white target.

2.3.2.1 Point Clouds

Using point clouds for navigation is simply a technique requiring no inherent reduction of raw perception data. In its raw form, the lidar data is mapped to global frame using pose of the robot when the lidar scan was taken. As future scans arrive, they can be compared to the current cloud via iterative closest point (ICP) or similar techniques to determine change in motion. It is also common, due to the significant memory requirements of storing large point clouds, to down-sample these point clouds or only store information over some given window of time.

Point clouds matching for navigation is demonstrated quite successively in [63] where an unmanned underwater vehicle (UUV) employs ICP to navigate the ocean floor. The authors specifically chose a point cloud representation for the environment due to the general lack of features in the operating environment. Additionally it is worth noting that the SLAM filter in [63] was formulated using an EKF approach. Because no truth data was available, and the exploration area was several thousand square meters, the truth metric employed was to analyze overlapping scans. Scans that did overlap appeared to correspond well with one another.

An approach with ground robots in point-cloud navigation is introduced in [28]. It is noted that multiple levels of relaxation at coarser and finer levels of resolution can lead to more efficient algorithms that can better handle the non-linearities of SLAM. Additionally, some benefit is shown through both simulated and real data that this approach also can aid in the closure of large loops on the order of 200m in length. The approach is similar to an occupancy grid approach where each point is projected from the lidar to determine areas assumed to be occupied or unoccupied. This is effectively a quantized point cloud, with some additional statistical information included. Such a technique was shown to be successful in [24] when navigating indoors with sonar.

While most ICP algorithms make an inherent assumption that the environment is static, SLAM with the aid of ICP has been shown to function in dynamic environments [43]. The

approach shown was to attempt to locate objects that were changing from scan to scan and essentially remove them from consideration. This approach was tested using a Velodyne laser scanner in both road traffic and stop-and-go traffic.

2.3.2.2 Lines and Arcs

Ideally SLAM can be performed in environments with more structure which allows features to be described geometrically, and therefore with fewer variables, which can reduce the overall computational burden of the algorithm. The most common feature descriptors are lines and arcs. These lines and arcs are found by analyzing the raw perception information, extracting the series of points that can be closely represented using these geometric descriptors, either separately or in conjunction with one another. Hence, instead of storing a series of points as with the point cloud method, the points are replaced by the equation used to describe the line or arc. However, the extraction process of the geometric features from the point clouds is a non-trivial process and has seen a swath of research [59, 20, 68, 67, 61].

A comparison of various line extraction methods and their implementations in the SLAM framework are analyzed in [60]. Specifically [60] analyzes these line based techniques in a series of office spaces with the longest corridors reaching 30 to 40 meters. While generally this approach shows good agreement with the provided map used as truth, errors can be on the tens of meters when long paths are taken that do not loop.

2.4 SLACAM Overview

Simultaneous localization auto-calibration and mapping (SLACAM), is a technique that combines SLAM with sensor auto-calibration techniques. Hence this technique attempts to determine the location of the robot, and map the environment using sensors that are known to not be properly calibrated, and during the course of performing SLAM, determine the correct alignment of these sensors. This technique is especially challenging because sensors are often calibrated using known information about the environment which is unavailable.

Additionally, SLAM assumes accurately calibrated sensors to navigate which will initially not be the case. Obviously the ability to perform SLACAM can greatly increase system robustness such that a robot with a jarred sensor can "heal" and overcome this impediment.

The need for auto-calibrated sensors is championed in [42], noting the difficulty and tedium associated with the calibration of 3D lidar given the vast quantity of information they supply. A generalized architecture and approach to the SLACAM problem is introduced in [27]. Additionally, [27] provides an overview for the implementation of SLACAM in both a centralized and federated Kalman filter frameworks. While it is noted that the centralized implementation is the optimal solution, the federated approach adds some ease of implementation with the ability to interchange various sensors into the greater SLACAM architecture. Additionally the author notes that if the number of calibration states is similar to the number of landmarks, a significant time savings could be realized with the federated approach. Additionally, an outline of a communication scheme is introduced so that a sensor could communicate with the filter when connected so that the filter can determine without user input what values need calibration.

One early work that incorporated techniques found in SLACAM is [84]. In [84] they are attempting to solve the simultaneous localization and tracking (SLAT) problem along with the calibration of their tracking sensors. The specific technique developed used Laplace's method to approximate the distribution of a Gaussian instead of more traditional Bayesian filtering techniques. Specifically the SLAT algorithm attempted to track the position of a cart moving through an environment where sensors were placed in a grid pattern of regular intervals. A sonar like ping was emitted from the cart, and when received by the tracking sensor would send the time stamp of the recorded ping to a central PC. From this information it was shown that the sensors could be calibrated and the cart could be tracked to within a few centimeters. Additional outdoor testing with a less dense sensor array increased positioning errors to nearly half a meter.

To reduce the complexity of the SLACAM problem, typically, the entire sensor suite is not assumed to have some error, and errors are assumed to be small. Generally, only one sensor is assumed to be in need of calibration, while other sensors on the platform remain calibrated. Below is an overview of several system that use SLACAM, where the subsections refer to the sensor or sensors on board they are attempting to calibrate.

2.4.1 Odometry Based SLACAM

A stochastic odometry calibration approach is presented in [76] where instead of explicitly trying to determine the correct kinematic parameters that optimized the robot trajectory, the optimal noise parameters are sought. This was accomplished during a SLAM routine where an occupancy grid was estimated and the error to be minimized was the current prediction of the environment verses the current map of the environment. This approach was validated through a number of real world datasets where the calibrated results showed a significant improvement over the initial estimates, often by eighty percent. Error was assessed by having the robot return to its initial location and determining the perceived location offset from the beginning with the true offset.

A federated filter approach to calibrate odometry values for the wheel radii and the longitudinal offset to a lidar while simultaneously attempting to locate itself, but not map, is presented in [48]. The calibration was performed by determining the change in pose of the lidar, estimating the change in distance transversed by each of the wheels, and comparing this to the measured change at the wheels. The calibrated values are then used in a separate EKF for the purpose of locating the robot itself. It was shown that this method was capable of estimating the error to within 5% of the total distance transversed by the robot.

An odometry auto-calibration scheme combined with SLAM is introduced in [49]. This technique separates the calibration and SLAM algorithms into two separate filters. This approach allows the auto-calibration filter to compare the odometry change in pose with the lidar change in pose to ascertain the effective wheel radii. These updated wheel radii are

then used in the SLAM algorithm to aid the position solution. This approach was validated by adding tape to the wheel of a robot and comparing the wheel radii calculated both with and without the tape.

2.4.2 IMU

The process of IMU alignment with another sensor or platform is of great interest in the aircraft community, a process known as bore-sight alignment. This is typically accomplished during higher dynamic maneuvers and comparing the results to GPS as seen in [73]. A similar technique was also shown to be feasible during lower dynamic maneuvers where the heading between an aircraft and IMU was calibrated during taxiing [74] by comparing the IMU with the change in position and velocity reported via the GPS to correct a two degree yaw offset. This correction was validated through the use of the flight profile data. A similar technique is presented in [70] analyzing the effect of using both EKF and UKF approaches. It was also shown that simply adding bearing only measurements from a camera effectively using structure from motion can significantly improve the performance of an IMU . This testing occurred using an aircraft, where the gimbaled camera system would compare images to aid in the determination of changes in positions. Note that these techniques are treating the IMU as the mal-aligned sensor, typically in reference to the body frame of the vehicle. This dissertation will treat the IMU as a fixed sensor to which other sensor will be calibrated, regardless of the alignment of the IMU to the body frame of the vehicle.

2.4.3 Lidar and Odometry

The relative pose between a lidar and the odometry of a system is determined while simultaneously calibrating the odometry values in [16]. The problem was treated as a constrained least squares problem where the change in pose of the lidar was determined via ICP and compared to the odometry values to find the relative pose between the sensors. Before

this could take place however, the lidar is simply used to aid in the calibration of the odometry values, in which the wheel radius and wheel base were determined. Experiments were conducted indoors and exhibited close agreement with the UMBmark [6] calibration results for the odometry values, and lidar results were within the accuracy of hand measurements.

A lidar and odometry calibration scheme is introduced in [37] that relies on solving the auto-calibration problem with least-squares and hyper-graphs. Specifically the SLAM problem is treated as a hyper-graph problem where each measurement acts as a node on a graph that contains pertinent position and sensor location parameters. Initially, the lidar undergoes calibration and is then held constant for the rest of the SLAM operations. The odometry by contrast only relies on the past fifty measurements to calibrate the effective wheel radii, and thus allowing for different surfaces to be quickly adapted to, at the risk of throwing out potentially beneficial information. The robot was tested by adding weight to alter the wheel radii as well as driving the robot on varying surfaces to see if a change in wheel radii could be detected. The method seen in [37] was also employed in [38] where it was found that despite the wheel radii remaining the same for both tile and carpet surfaces, the variance on the radii measurement increased, thus providing a future to detect surface variations.

The observability of the SLAC process is analyzed in [50]. Essentially a number of different scenarios are analyzed and it is determined that SLAC is observable for a lidar and differential drive robot when the distance to an object and the absolute robot orientation are known. Optimal calibration paths are also analyzed for a holonomic vehicle where it is noted that optimal paths occur essentially when the acceleration in the x and y directions is maximized.

2.4.4 Lidar and IMU

While not strictly lidar, a vision to IMU technique is demonstrated in [34]. This technique calibrates the two sensors using a checkerboard target, and relying on only one axis

being excited at a time. The results presented where from a laboratory test where the relative pose was found to within 0.5cm and approximately 0.1°.

A generic calibration technique based on manifolds is introduced in [88]. This technique is based on manifolds which are solved with nonlinear least squares and demonstrates its ability to calibrate a camera to an IMU using incremental pose measurements. Additionally it is shown that it is capable of finding intrinsic calibration parameters such as the those for a pin-hole model of a Microsoft Kinect. Additionally, it is noted that this technique has the potential to be used to solve the SLAM problem. While [88] lacks extensive testing it does provide a number of different scenarios where it was employed.

Another generic calibration technique is introduced in [12] where sensor to sensor calibration is performed by using incremental pose information. Essentially, the relative change in pose of each sensor is calculated as the robot maneuvers. The calibration parameters are then found by solving a cost function with least squares. In addition to demonstrating the performance of the algorithm in 2D, an observability analysis is performed and various degenerate paths are outlined. The work in [12] is further expanded in [13], where the algorithm is modified to function in three dimensions. The validation of this algorithm was performed by calibrating two lasers to one another.

An odometry calibration scheme is introduced in [95] where a vision and IMU system is combined to estimate slip parameters of a skid steered robot. Effectively these slip parameters are scale factors to the wheel radius that are allowed to change over some bounded area. The author demonstrates that when these parameters are allowed to be accounted for, the overall odometry solution is improved. The results of [95] are further verified in [94] where longer paths of varying shape and surface are tested. It is worth noting that tile surfaces appeared to experience less slip such thus allowing the calibration to perform slightly better on than concrete or sand.

The odometry of a differential drive robot is calibrated as well as the lidar mounting location in [16]. ICP is employed to compare the relative pose estimates between the lidar

and the odometry. These results allow for the distance between the wheels of the robot, radii, and mounting location of the lidar to be determined using least squares. The authors noted that there were timing and other instrument and measurement errors that occurred which were removed and compensated for simply through the use of a wealth of data.

2.5 Conclusion

This chapter has provided an overview of the state of the art of both auto-calibration and it progression into SLACAM by combining it with SLAM. Additionally it was shown that most auto-calibration techniques are cumbersome and are specific to the sensor being calibrated. Those techniques that are more robust to different sensor types have not demonstrated anything more that in laboratory testing and generally lack rigorous validation. Additionally SLACAM techniques by and large only calibrate a single sensor, and in the rare event of multiple sensors being calibrated the implementation is sub-optimal where only a small window of data is analyzed for calibration while the other sensor to calibrate is held constant. This dissertation builds on and advances the state of the art by introducing a sensor calibration technique based on incremental poses which is analyzed in a real-world environment and whose calibration result is independently validated. Furthermore a SLACAM technique is implemented that calibrates an IMU to an odometry and lidar system. This calibration scheme is implemented using an EKF such that all sensor information is used for all aspects of the calibration, navigation, and mapping scheme.

Chapter 3

Auto-Calibration

This chapter will present an auto-calibration technique that relies heavily on the ability of sensors to obtain their relative motion. Therefore after the general calibration technique is presented, a detailed explanation of the raw sensor measurements and relative motion estimation strategies for each of the core sensors to be calibrated will be presented. These techniques will form the foundation for the auto-calibration testing preformed in Chapter 4 and the basis of the auto-calibration portion of SLACAM in Chapter 5. Note that the presented technique inherently seeks to determine the extrinsic calibration between two sensors, that is the physical relationship between two sensors, such as translational or rotational offsets. These typically desired calibration parameters are shown in Figure 3.1, where v is the reference frame, and the frame denoted as s is the frame to be calibrated. Hence, the desired parameters are the rotation from the reference frame to the sensor frame denoted as R_v^s as well as the translation from the reference frame to the sensor frame denoted as S_o^v.

This is different from intrinsic calibration which seeks to determine some inherent value about the sensors itself, such as noise levels, scale factors, or non-orthogonal axes. Also note that this calibration technique is specifically a sensor to sensor calibration technique, and not inherently a sensor to body frame calibration technique. Theses extrinsic calibration values are typically the exact same values used by the navigation algorithm to process sensor measurements. Incorrect extrinsic values can cause minor problems from merely sub-optimal navigation performance to major problems of total filter divergence.

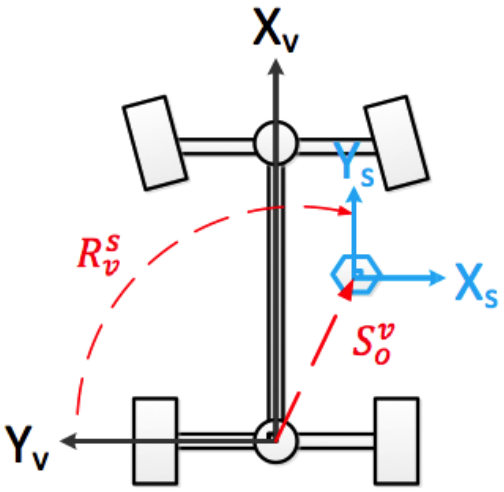

Figure 3.1: Desired Calibration Parameters

3.1 General Technique

Typically extrinsic calibration techniques take one of two approaches. The first approach is to rely on sensors sharing overlapping environmental observations, and the second approach relies on sensors sharing dynamics which are typically known a-priori. This first technique is popular amongst lidar and camera calibration techniques, where observed information of the environment from each sensor is compared to one another. This is typically accomplished by extracting features from both measurements and attempting to find rotation and translation parameters that allow the two observations to most closely align, as seen in [45, 5]. The second technique is more common among researchers attempting to calibrate sensors to an IMU. Hence a typical assumption is that the dynamics are isolated to one axis so that the problem can be constrained and any dynamics observed in the other axes can be corrected for by simply finding the sensor alignment that best fits the aforementioned assumption as seen in [97, 98]. Regardless of the technique, it is critical for the two sensors in need of calibration be capable of making observations which can be compared to one another. The downside of these two approaches is that while they have the potential to yield satisfactory results, they lack interchangeability between sensors. Hence for a given system employing

multiple sensors, multiple calibration techniques must be employed to calibrate the entire system. This is time consuming, tedious, and often impractical for a user in the field.

However a technique was first presented in [12] for the 2D case based on relative pose measurements to reduce the assumptions on the sensors being used as well as relies on more reasonable and realistic assumptions of dynamics. It is the technique presented in [12] which is expanded to the 3-D case by this dissertation and forms the basis of this dissertations auto-calibration strategy. The technique presented in this dissertation assumes that each sensor is capable of reporting a relative change in pose - position and attitude, and that the sensors are rigidly mounted on the same platform. Additionally the technique presented here assumes that the platform to which the sensors are mounted experiences excitation in all axes. Essentially the premise behind this technique is that if two sensors are perfectly aligned they will measure the same relative pose, and if they are not aligned they will not measure the same relative pose, but an optimal translation and/or rotation can be found to align the two reference frames. Therefore because this technique relies on relative pose, a broad range of sensors are capable of being calibrated. Note that the sensor does not need to inherently measure relative pose such as a lidar, but simply must be capable of deriving this measurement from its own observation information.

As previously mentioned, and shown in Figure 3.1, the typical sensor to sensor calibration parameters are R_v^s and S_o^v. Specifically, S_o^v represents the origin of the sensor to be calibrated in the reference frame, and is comprised of the variables X_o, Y_o, and Z_o which simply represent the offset in the given cardinal axis. Thus S_o^v is defined in Equation (3.1).

$$S_o^v = \begin{bmatrix} X_o & Y_o & Z_o \end{bmatrix}^{\mathrm{T}} \tag{3.1}$$

Similarly, the rotation from the reference frame to the sensor to be calibrated is defined by its Euler angles. Specifically these are a yaw offset denoted as ψ, a pitch offset denote as θ, and roll offset denoted as ϕ. Thus the rotation matrix R_v^s is defined in Equation (3.2).

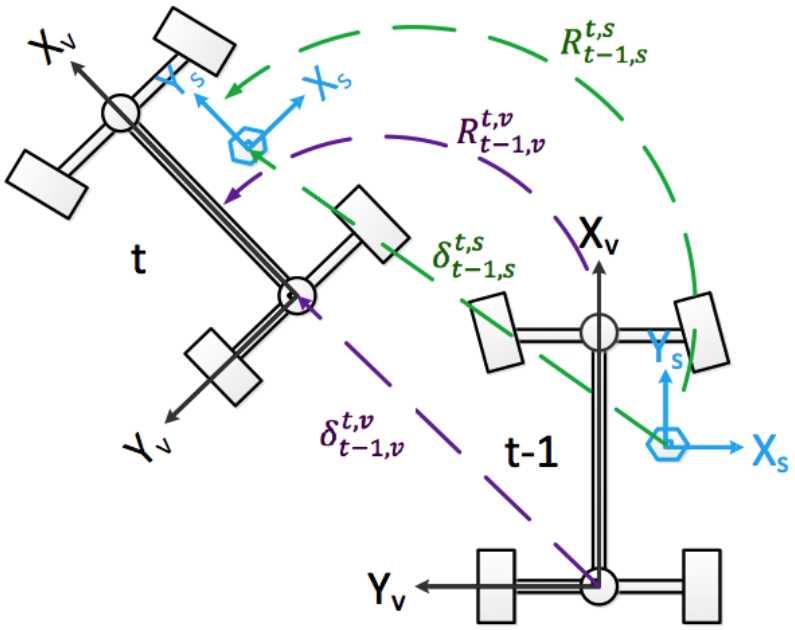

Figure 3.2: Previous to Current Translation and Rotations

$$R_v^s = (R_s^v)^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Determining the relative motion of each sensor is critical to the calibration technique. Specially, the required values for the reference frame are $R^{t,v}_{t-1,v}$ and $\delta^{t,v}_{t-1,v}$ which represent the rotation and translation from the previous frame to the current frame. Similarly for the sensor frame, the required values are $R^{t,s}_{t-1,s}$ and $\delta^{t,s}_{t-1,s}$ which represent the rotation and translation from the previous frame to the current frame of the sensor to be calibrated. These values can be seen in Figure 3.2.

The translation and rotation parameters are defined in Equations (3.3) and (3.4),

$$\delta_{t-1,f}^{t,f} = \begin{bmatrix} \delta_{x,f}^{f} & \delta_{y,f}^{f} & \delta_{z,f}^{f} \end{bmatrix}^{\mathrm{T}} \tag{3.3}$$

$$\begin{aligned} R_{t-1,f}^{t,f} = & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_f) & \sin(\phi_f) \\ 0 & -\sin(\phi_f) & \cos(\phi_f) \end{bmatrix} \\ & \begin{bmatrix} \cos(\theta_f) & 0 & -\sin(\theta_f) \\ 0 & 1 & 0 \\ \sin(\theta_f) & 0 & \cos(\theta_f) \end{bmatrix} \begin{bmatrix} \cos(\psi_f) & \sin(\psi_f) & 0 \\ -\sin(\psi_f) & \cos(\psi_f) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.4}$$

where the symbol f can be substituted for either v or s to denote the motion in the appropriate frame. Note that ϕ_f, θ_f, and ψ_f represent the relative roll, pitch, and yaw rotations experienced by their receptive frames. Now that the equations for the relative motion exist for the reference and sensor frames, it is simply a matter of selecting an optimization technique to ascertain the desired parameters.

3.2 Extended Kalman Filter Using Relative Pose

Solving for the mounting parameters using an extended Kalman filter (EKF) is often ideal. This formulation allows for the method to be added with ease to a pre-existing filter so that calibration can take place in parallel to other navigation and estimation tasks. In order to estimate the calibration parameters using relative pose measurements of the sensor location, a prediction of its movements need to be described using only the reference sensor and the current best estimate of the mounting parameters as presented in Equation (3.5).

$$\begin{aligned} \hat{\delta}_{t-1,s}^{t,s} &= R_v^s \left(\delta_{t-1,v}^{t,v} + R_{t-1,v}^{t,v} S_\circ^v - S_\circ^v \right) \\ \hat{R}_{t-1,s}^{t,s} &= R_v^s R_{t-1,v}^{t,v} R_s^v \end{aligned} \tag{3.5}$$

Thus any error between the predicted change in motion and the measured change in motion will be interpreted as a mounting error. Therefore, the states that will be used in the EKF are shown in Equation (3.6) and are the values that are necessary to construct the $S_{\circ}^{v}$ and R_{v}^{s} matrices.

$$\mathbf{X} = \begin{bmatrix} X_{\circ} & Y_{\circ} & Z_{\circ} & \psi & \theta & \phi \end{bmatrix}^{\mathrm{T}} \tag{3.6}$$

The process and measurement models for the EKF are presented in Equation (3.7) and are modeled as being subjected to Gaussian white noise.

$$\begin{aligned} \mathbf{X}_{k+1} &= \Phi_k \mathbf{X}_k + \mathbf{w}_k \quad \mathbf{w}_k \sim N(0, Q_k) \\ \mathbf{z}_k &= H_k \mathbf{X}_k + \mathbf{v}_k \qquad \mathbf{v}_k \sim N(0, R_k) \end{aligned} \tag{3.7}$$

The mounting parameters are modeled as remaining effectively constant throughout time. Determining if the mounting location of a sensor has changed during auto-calibration is explored in Chapter 4. However, because the states are modeled as remaining effectively constant, Φ_k reduces to the identity matrix. This has the affect of having the covariance of the time update being effectively driven by white noise as seen in Equation (3.8).

$$P_k^- = P_{k-1} + \mathrm{diag}(\mathbf{w}_k) \tag{3.8}$$

The linearized measurement matrix H_k is defined in Equation (3.9)

$$H_k = \frac{\partial}{\partial \mathbf{X}_k} G\left(\delta_{t-1,v}^{t,v},\ R_{t-1,v}^{t,v},\ \mathbf{X_k}\right) \tag{3.9}$$

where the nonlinear measurement matrix is defined in Equation (3.10),

$$\begin{bmatrix} \hat{\delta}^{t,s}_{t-1,s} \\ \hat{\psi}_s \\ \hat{\theta}_s \\ \hat{\phi}_s \end{bmatrix}_k = G\left(\delta^{t,v}_{t-1,v},\ R^{t,v}_{t-1,v},\ \mathbf{X_k}\right) \tag{3.10}$$

is a function of the current mounting parameters ($\mathbf{X_k}$) and the change in translation and rotation of the vehicle from the previous frame to the current frame ($\delta^{t,v}_{t-1,v}$ and $R^{t,v}_{t-1,v}$) as defined in Equations (3.3) and (3.4) and predicts the incremental translation and rotation from the previous frame to the current frame experienced by the sensor frame. Specifically, Equation (3.10) is calculated using Equation (3.11) where $\hat{\delta}^{t,s}_{t-1,s}$ and $\hat{R}^{t,s}_{t-1,s}$ have been calculated using Equation (3.5).

$$G = \begin{bmatrix} \begin{bmatrix} \hat{\delta}^{t,s}_{t-1,s} \end{bmatrix} \\ \arctan 2\left(\frac{\hat{R}^{t,s}_{t-1,s}(2,1)}{\hat{R}^{t,s}_{t-1,s}(1,1)}\right) \\ \arctan 2\left(\frac{-\hat{R}^{t,s}_{t-1,s}(3,1)}{\sqrt{\left(\hat{R}^{t,s}_{t-1,s}(3,2)\right)^2+\left(\hat{R}^{t,s}_{t-1,s}(3,3)\right)^2}}\right) \\ \arctan 2\left(\frac{\hat{R}^{t,s}_{t-1,s}(3,2)}{\hat{R}^{t,s}_{t-1,s}(3,3)}\right) \end{bmatrix} \tag{3.11}$$

Note that $R(r, c)$ denotes the row and column of the given rotation matrix and $\arctan 2$ denotes the four quadrant inverse tangent. Thus the innovation of the EKF is seen in Equation (3.12), is the difference between the measured change in motion of the sensor frame and the predicted change in motion of the sensor frame. The innovation is also the value that is sought to be minimized.

$$y_k = z_k - \begin{bmatrix} \hat{\delta}^{t,s}_{t-1,s} \\ \hat{\psi}_s \\ \hat{\theta}_s \\ \hat{\phi}_s \end{bmatrix}_k \tag{3.12}$$

Through the calculation of the Kalman gain seen in Equation (3.13),

$$K_k = P_k^- H_k^{\mathrm{T}} \left(H_k P_k^- H_k^{\mathrm{T}} + R_k \right)^{-1} \tag{3.13}$$

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_{\mathbf{k}}^- + K_k y_k \tag{3.14}$$

an updated mounting prediction as well as an update estimate covariance can be calculated as seen in Equations (3.14) and (3.15).

$$P_k = \left(I - K_k H_k \right) P_k^- \tag{3.15}$$

Thus, the equations have been developed to predict the change in motion of the sensor from the previous frame to the current frame using only measurements from the vehicle frame and the predicted calibration parameters. Note that this algorithm makes no inherent assumptions on the type of sensor used, merely its ability to produce a relative pose estimate. Now, it is simply a matter of determining the relative pose from each sensor to be calibrated.

3.3 Lidar Based Relative Pose

A lidar is a sensor that fundamentally makes range and reflectivity measurements. Because there is no raw output of change in pose of the lidar, it must be inferred from the raw measurements. The most common method of obtaining relative pose from a lidar is a technique introduced in [17] and [4] known as iterative closest point (ICP) which compares successive lidar measurements and infers the change in pose of the lidar. The fundamentals

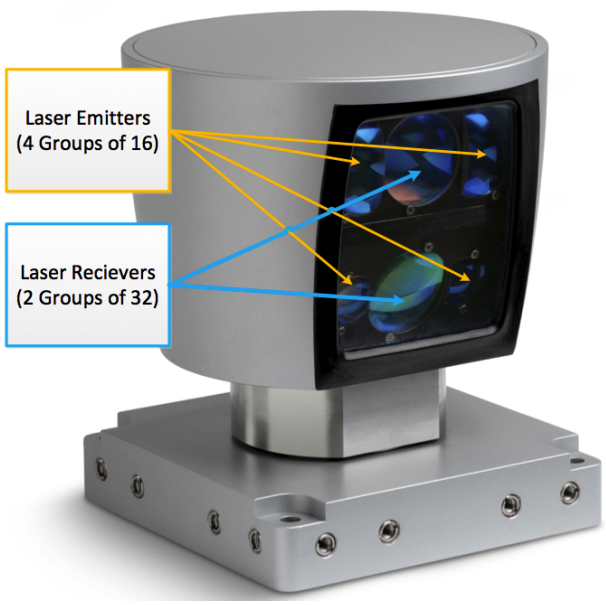

Figure 3.3: 3D Velodyne Lidar

of lidar based measurements are introduced in 3.3.1, and a more detailed outline of ICP is provided in 3.3.2.

3.3.1 Lidar Sensor Fundamentals

A lidar is a device that uses the principle of time of flight to make range measurements. This is typically achieved by either having multiple lasers and receptors rotating about a common axis, such as the Velodyne lidar as shown in Figure 3.3 which is capable of measuring in all three axes. Alternatively, most 2D lidars rotate simply a mirror that directs the beam to a known angular measurement which makes the typical fan shaped scan pattern as seen in Figure 3.4. Note that based on the strength of the return pulse, the reflectivity of an object can also be measured.

Because of the manner in which measurements are made by the lidar, the polar coordinate system is typically used to report lidar measurements. As seen in Figure 3.5, the range of the lidar measurement is defined as ρ, the vertical angle if one exists of the lidar measurement is defined as α, and the horizontal rotation value is defined as γ.

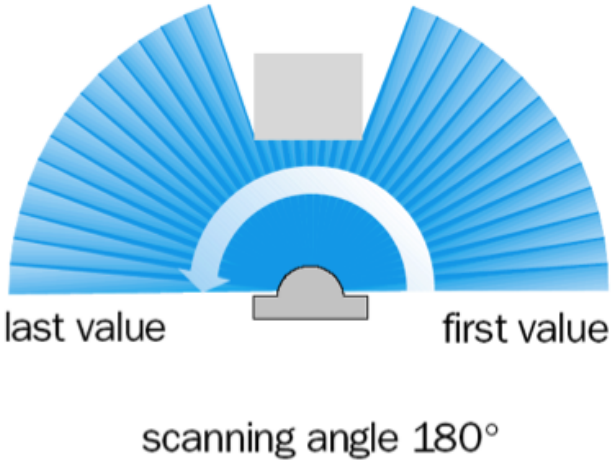

Figure 3.4: 2D Lidar [89]

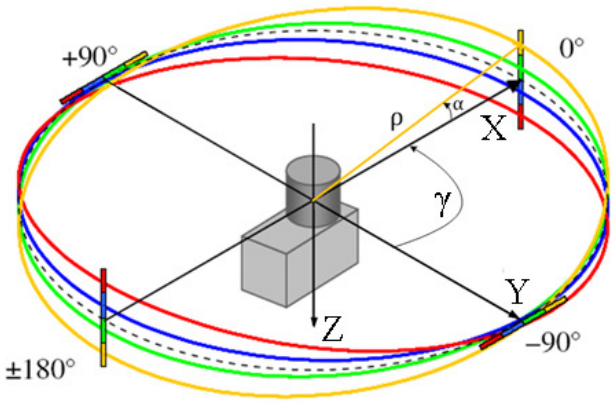

Figure 3.5: Lidar Coordinate Frame [32]

3.3.2 Obtaining Relative Pose via ICP

Iterative Closest Point (ICP) is a broad technique that can determine the translation and rotation of the scene a lidar is measuring. While not limited to determining environmental motion, it is one of ICPs most common uses. The standard implementation of ICP assumes that the algorithm is provided with two point clouds taken at different times in a static environment. One scan is treated as the model and the other is treated as the source, where the model is typically the initial scan, and the source is the scan which needs to be compared to the model to determine change in pose. In its original form, the closest points between the model and source scans are determined, and the translation and rotation from scan to scan is found by recursively minimizing this distance. Note that new corresponding points are determined with each iteration, and it is common to provide the algorithm with an initial estimate of motion. A demonstration of the process is presented in Figure 3.6, where two scans have been aligned which where taken several meters apart. Note the alignment of the straight walls specifically the the one located at the -350m location in the Y axis. It is worthwhile to also note that there have been a number of techniques to increase the speed and robustness of this technique, namely that of the use of K-D trees [80], and the use of point to line or point to plane minimization instead of simply point to point minimization [44, 78].

3.4 Inertial Based Relative Pose

Obtaining relative pose from inertial based measurements is a relatively straight forward process. An inertial measurement unit (IMU) typically consists of a triad of accelerometers (accels) and gyroscopes (gyros) mounted orthogonally to one another allowing measurements to be made in all three cardinal axes. From the raw measurements of the sensors, change in rotation, position, and velocity can be derived. The major benefit of using this type of sensor, is that it can produce these change in pose measurements with no outside aiding. A

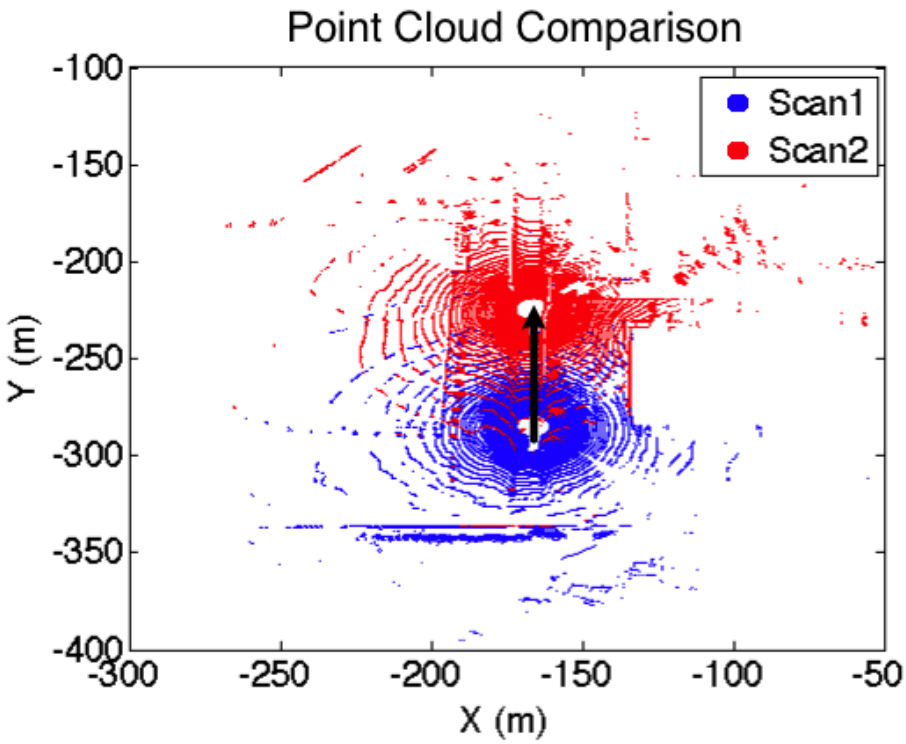

Figure 3.6: Successive Point Clouds

brief overview of the sensor fundamentals, and necessary equations for modeling the sensor are provided below.

3.4.1 IMU Sensor Fundamentals

Fundamentally, an IMU is a type of dead-reckoning sensor that produces measurements of specific force from the accelerometers (accels) and angular rate from the gyroscopes (gyros). Velocity is obtained by integrating the specific force measurement over the sample period, and change in position is obtained from integrating the velocity result with respect to time. Similarly, the change in attitude can be obtained by integrating the angular rate information. Typically, for this information to be of use for navigation, an initial position and attitude must be provided. Modern IMUs operate in a "strap-down" configuration, such that they are rigidly connected to the object being sensed, and the measurements are integrated via computer to determine a navigation solution. There are varying grades of IMUs:

navigation grade, tactical grade, and microelectromechanical systems (MEMS) grade. These grades often denote the drift rate of the gyros and are directly proportional to the cost of the IMU. The equations necessary to transform the inertial measurements into changes in position and attitude are provided below for navigation in the local frame.

The IMU provides angular rate measurements denoted as ω_{ib}^{b}. These measurements can be used to update the current attitude matrix denoted as C_{b}^{e} which is a rotation from the body to the Earth Centered Earth Fixed (ECEF) frame as denotes in Equation (3.16).

$$\dot{C}_{b}^{e} = C_{b}^{e}\Omega_{eb}^{b} = C_{b}^{e}\Omega_{ib}^{b} - \Omega_{ie}^{e}C_{b}^{e} \tag{3.16}$$

Where "e" denotes the ECEF frame, "b" denotes the body frame, and "i" denotes the inertial frame. Note that Ω_{ib}^{b} is the skew symmetric matrix of the IMU's angular-rate measurement, and Ω_{ie}^{e} is the skew-symmetric matrix of the Earth-rotation vector which is assumed to remain constant. Alternatively, the updated attitude matrix ($C_{b}^{e}(+)$) can be approximated using the previous attitude matrix ($C_{b}^{e}(-)$) via Equation (3.17) when Equation (3.16) has been integrated over an integration period of τ_i.

$$C_{b}^{e}(+) \approx C_{b}^{e}(-)\left(I_3 + \Omega_{ib}^{b}\tau_i\right) - \Omega_{ie}^{e}C_{b}^{e}(-)\tau_i \tag{3.17}$$

The acceleration of the body is denoted in Equation (3.18).

$$\dot{v}_{eb}^{e} = f_{ib}^{e} + g_{b}^{e}\left(r_{eb}^{e}\right) - 2\Omega_{ie}^{e}v_{eb}^{e} \tag{3.18}$$

The specific force measurement f_{ib}^{e} from an ideal IMU resolved in the ECEF frame whose calculation is presented in Equation (3.19).

$$f_{ib}^{e}(t) \approx \frac{1}{2}\left(C_{b}^{e}(-) + C_{b}^{3}(+)\right)f_{ib}^{b} \tag{3.19}$$

where $g_b^e\left(r_{eb}^e\right)$ is the local gravity model. Thus, the updated velocity term is presented in Equation (3.20).

$$v_{eb}^e(+) \approx v_{eb}^e(-) + \left(f_{ib}^e + g_b^e\left(r_{eb}^e(-)\right) - 2\Omega_{ie}^e v_{eb}^e(-)\right)\tau_i \tag{3.20}$$

Finally, the updated position can be calculated using Equation (3.21):

$$r_{eb}^e(+) = r_{eb}^e(-) + \left(v_{eb}^e(-) + v_{eb}^e(+)\right)\frac{\tau_i}{2} \tag{3.21}$$

However, the above set of equations can be simplified for some scenarios. For example, if the robot is constrained to level ground a two dimensional earth can be assumed reducing the problem to three degrees of freedom. Additionally, with a MEMS grade IMU, it is possible to assume a non-rotating earth without serious detriment to the navigation solution due to the fact that the noise and drift on the IMU is so great that this phenomena is not easily observed. Hence, the state update equations could also be written as seen in Equations (3.22 - 3.25), where the raw measurements from the IMU are denoted as they where in Equations (3.16 - 3.21).

$$\dot{v}_{nb}^n = C_b^n f_{ib}^b \tag{3.22}$$

$$v_{nb}^n(+) = v_{nb}^b(-) + f_{ib}^e\tau_i \tag{3.23}$$

$$r_{nb}^n(+) = r_{nb}^n(-) + \left(v_{eb}^e(-) + v_{eb}^e(+)\right)\frac{\tau_i}{2} \tag{3.24}$$

$$C_b^n(+) \approx C_b^n(-)\left(I_3 + \Omega_{ib}^b\tau_i\right) \tag{3.25}$$

Figure 3.7: Applanix INS [10]

Note that these equations are also resolved in the local navigation frame, which has been denoted via "n" in both subscript and superscript.

3.4.2 Inertial Navigation Systems

Inertial Navigation Systems (INS) are systems based around an IMU, that blend other additional sensors in an effort to determine either relative changes in position or global pose. Alternatively, and INS can simply mean an integrated IMU with no additional blending, however this is not the definition adopted in this dissertation. Often these additional sensors include GPS, wheel odometry, and magnetometers. An Applanix POS-LV 220, is an example of such an INS, and is shown in Figure 3.7. Note that this system blends an IMU, wheel odometry, and GPS, which it uses to report a change in pose solution. However, this system does not inherently allow access to all of the raw measurements that each individual sensor is capable of measuring.

3.5 Wheel Odometry Based Relative Pose

Because of the simplicity in using wheels as a form of locomotion, they are often the defacto means of ground robot propulsion. Wheel based odometry seeks to determine the change in pose from some initial starting location by determining how far each wheel has rotated combined with the known robot geometry. While wheel odometers are extremely prevalent due to both cost and simplicity, they can suffer from serious navigation errors when the wheels freely spin on slick surfaces or when the robot slides when attempting to stop (known as wheel slip).

3.5.1 Wheel Odometry Sensor Fundamentals

A wheel odometer at its core simply counts transitions on an encoder disc attached to a robots transmission, often the axle or wheel itself. These transitions are often detected using light or Hall-effect. Encoder discs as seen in Figure 3.8, can be constructed such that the direction that the wheel is spinning can be determined so that it can be inferred whether the robot is moving forward or in reverse. The odometers used in this dissertation provided a count of encoder tics where if the encoders wheel was moving forward, the encoder count incremented and decremented if rotating in reverse. Additionally, these odometers are already compensated for which side of the vehicle it was located, such that the rotation rate of the left and right wheels, denoted ω_l and ω_r can be calculated via Equation (3.26).

$$\omega_l = \omega_r = \frac{Odom(t) - Odom(t-1)}{tpr} \frac{2\pi}{dt} \tag{3.26}$$

Note that the respective left or right odometer readings come from their respective $Odom(t)$ and the total number of ticks-per-revolution of the wheel encoder is denoted as tpr.

While it is common to model multi-wheel skid-steer vehicles as simply having two effective wheels located in the center of the robot as denoted noted by two green ovals in Figure 3.9. It has been shown in [93, 46, 52, 51] that this is an inaccurate model.

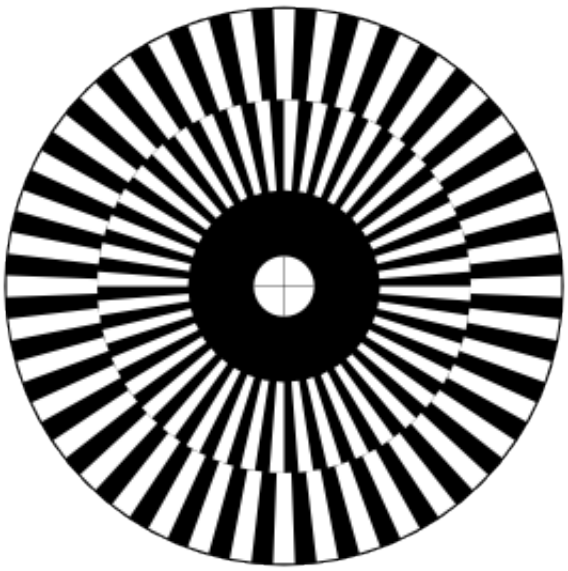

Figure 3.8: Wheel Encoder Disc [30]

Therefore, a more accurate kinematic representation of the skid-steered mobile robot is put forth that denotes that the robot should be modeled as rotating about some instantaneous center of rotation (ICR_v) where the left and right wheel pairs are still modeled as a single wheel but at locations ICR_l and ICR_r as seen in Figure 3.10. Theoretically the locations of ICR_l and ICR_r are constant, but practically remain in a bounded area located between each wheel pair in the X direction, and typically extend further than the wheel in the Y direction. Additionally, the radius of the wheel is not inherently representative of the true wheel dimensions, but is a value that accounts not only for the wheel radius, but also gear ratios, and for slip, such that this radius can be different for different surfaces the robot is driven on. Also note, that the X value denoted x_{ICR} for both ICR_l and ICR_r will be the same value, however the Y value denoted y_{ICR_l} or y_{ICR_r} can be different depending on if the center of mass favors one side in addition to environmental factors.

Thus, the equation describing the velocities and rotation rate of the robot in the body frame given the wheel rotation rates ω_l, ω_r, and the necessary ICRs is provided in Equation (3.27).

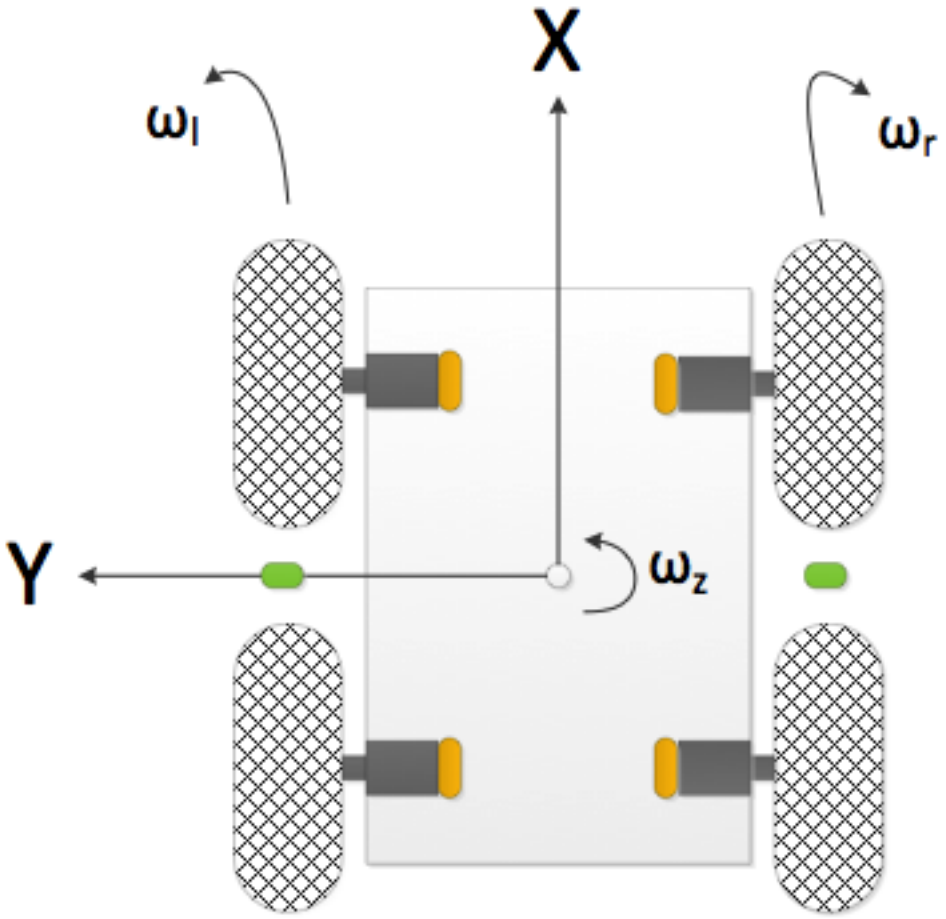

Figure 3.9: Poor Approximate Wheel Location

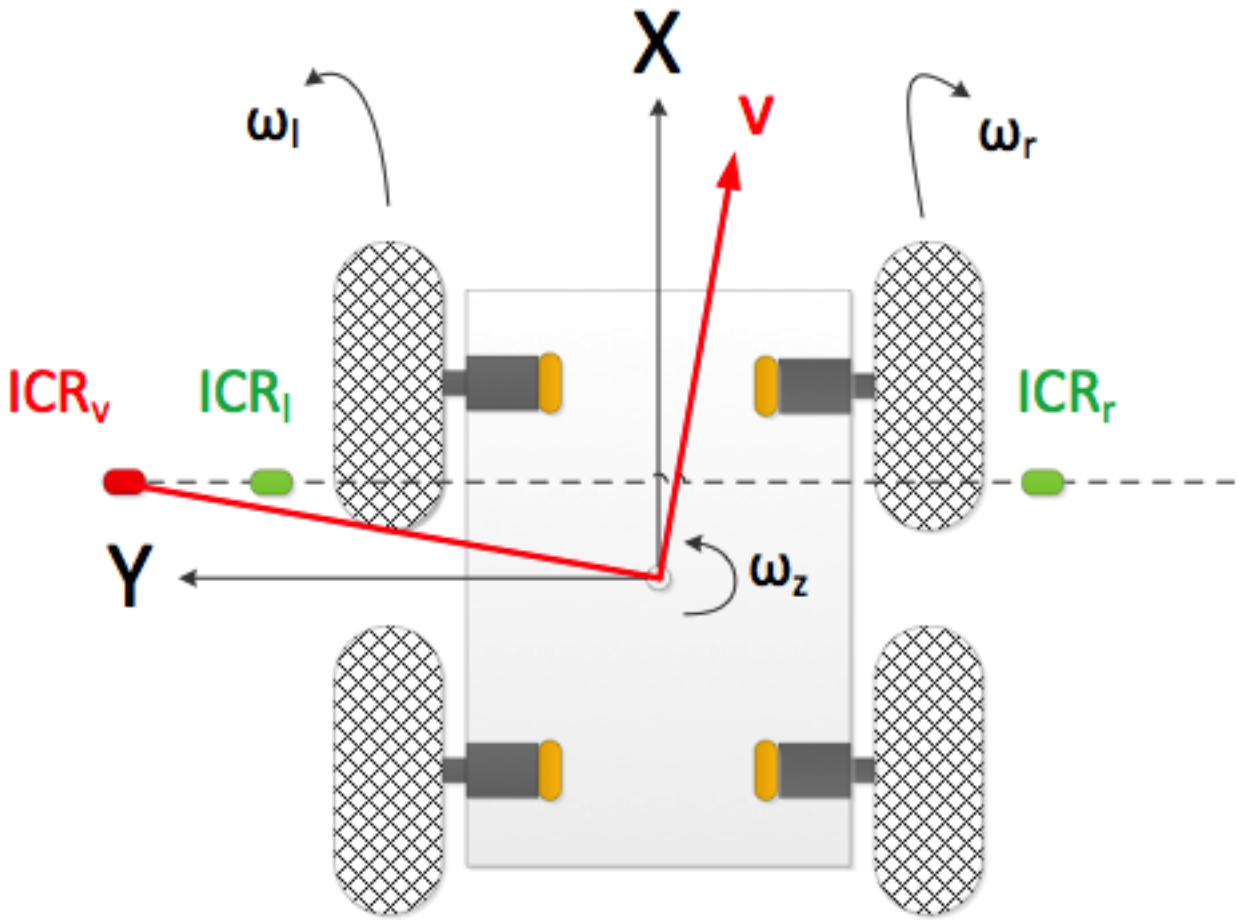

Figure 3.10: Desired Odometry Values

$$\begin{bmatrix} v_x^b \\ v_y^b \\ \omega_z \end{bmatrix} = \frac{1}{(y_{ICR_l} - y_{ICR_r})} \begin{bmatrix} -R_l y_{ICR_r} & R_r y_{ICR_l} \\ -R_l x_{ICR} & R_r x_{ICR} \\ -R_l & R_r \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{3.27}$$

From Equation (3.27) the necessary parameters are obtained to use the robot's odometry are the location of the ICR_l and ICR_r as well as the wheel radii for the left and right wheels denoted R_l and R_r. However, these values cannot be easily determined a-priori, and some type of calibration procedure must be performed to obtain them. Naturally, if the change in pose is desired, it is simply a matter of multiplying the velocity and rotation rates by the sample period. Additionally to determine the velocities in the navigation frame, it is simply a matter of rotating them by the current robot heading about the robots Z axis, if a 3DOF case is assumed.

3.6 Observability

If the output history, $z(t)$, $t_\circ \leq t \leq t_f \leq \infty$ is sufficient to reconstruct the state vector at the initial time ($x(t_\circ)$) then a system is said to be observable [83]. Therefore, before the calibration parameters are estimated, it must first be shown that there is sufficient information to successfully estimate the desired values. The observability of the system will be found using the Fischer Information Matrix (FIM) as derived by the Cramér Rao lower bound (CRLB). Final observations on observability and scenarios to avoid are addressed in 3.6.3.

3.6.1 Cramér Rao Lower Bound

The CRLB introduced by Herald Cramér and Calyampudi Rao is a means of determining the lower bound on the variance of a value to be estimated [18, 71, 85]. Hence if some value x is being estimated using observations $\hat{x}$ corrupted by some Gaussian noise (v) such that:

$$\hat{x} = x + v, \;\; v = N(0, \sigma_x^2) \tag{3.28}$$

Then the covariance $E\left((\hat{x} - x)(\hat{x} - x)^{\mathrm{T}}\right)$ can be calculated which will always be greater than or equal to the reciprocal of the FIM J:

$$E\left((\hat{x} - x)(\hat{x} - x)^{\mathrm{T}}\right) \geq J^{-1} \tag{3.29}$$

Thus Equation (3.29) provides a method to predict how well the states can be constrained. Hence if J^{-1} is not invertible, this implies that the variance of at least one parameter is unbounded and that J is not full rank. Therefore, if J is rank deficient the mounting parameters cannot be estimated, meaning the states are not observable. Situations on where this can occur are discussed in greater detail in Section 3.6.3.

3.6.2 Fischer Information Matrix

The iterative FIM for an EKF will adopt similar notation and methodology as described in [85]. The FIM requires that only the model and measurement uncertainties are known and that the estimator is unbiased as denoted in [92, 90]. Then the FIM can be written as:

$$\mathbf{J}_k = -E\left[\frac{\partial^2 \ln p_k}{\partial x_k^2} |\; \mathbf{X}_k\right] \tag{3.30}$$

where p_k is the conditional density of $\mathbf{Z}_k$ such that

$$p_k = p_{\mathbf{Z}_k|\mathbf{x}_k} \tag{3.31}$$

where

$$\mathbf{Z}_k = [\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k] \tag{3.32}$$

and

$$\mathbf{x}_k = [\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_k] \tag{3.33}$$

Equation (3.30) can then be written recursively as:

$$J_k = \left(\Phi_{k-1}^{-1}\right)^{\mathrm{T}} J_{k-1} \Phi_{k-1}^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \tag{3.34}$$

Noting that Φ_k was the identity matrix, then Equation (3.34) will reduce to the following:

$$J_k = J_{k-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \tag{3.35}$$

Therefore for J_k to be full rank the matrix $\left[H_k^{\mathrm{T}}, H_{k-1}^{\mathrm{T}}, \ldots, H_0^{\mathrm{T}}\right]^{\mathrm{T}}$ must also be full rank. Hence:

$$\operatorname{rank}(J_k) = \operatorname{rank}\left(\left[H_k^{\mathrm{T}}, H_{k-1}^{\mathrm{T}}, \ldots, H_0^{\mathrm{T}}\right]^{\mathrm{T}}\right) \tag{3.36}$$

Thus we see that the Jacobian of our measurement matrices must be full rank in order for the system to be observable.

3.6.3 Observability Conditions

Because J_k is a rather large and complex matrix, the observability of the system was analyzed using both numeric and symbolic manipulations to determine when the system became unobservable. As denoted in Equation (3.36), the Jacobian of the measurement matrix (H) must be full rank for the system to be observable. Similar to the 2D system presented in [12], one observation (from both the reference sensor and the sensor to be calibrated) does not contain sufficient information to estimate the calibration parameters as denoted by J_k being rank deficient as proven in Appendix A. However, if two or more observations are made for a given non-pedantic path, the system will be observable. It is worthwhile to note that once the system has achieved full rank the system will remain full

rank. Because new observations will be exponentially weighted, the following non-observable cases should still be avoided despite already obtaining sufficient information to estimate the desired values. All non-observable conditions can be avoided if each axis observes a non-constant change in attitude. The scenario in which the attitude never changes $\left(R^{t,v}_{t-1,v} = 0\right)$ is described in Equation(3.37).

$$\begin{gathered}\hat{\delta}^{t,s}_{t-1,s} = R^s_v\left(\delta^{t,v}_{t-1,v} + R^{t,v}_{t-1,v}S^v_\circ - S^v_\circ\right) = R^s_v\left(\delta^{t,v}_{t-1,v} - S^v_\circ\right) \\ \hat{R}^{t,s}_{t-1,s} = R^s_v R^{t,v}_{t-1,v} R^v_s = 0\end{gathered} \tag{3.37}$$

Which then implies Equation(3.38) for the nonlinear measurement.

$$G = \begin{bmatrix} \left[\hat{\delta}^{t,s}_{t-1,s}\right] \\ \arctan 2\left(\frac{\hat{R}^{t,s}_{t-1,s}(2,1)}{\hat{R}^{t,s}_{t-1,s}(1,1)}\right) \\ \arctan 2\left(\frac{-\hat{R}^{t,s}_{t-1,s}(3,1)}{\sqrt{\left(\hat{R}^{t,s}_{t-1,s}(3,2)\right)^2 + \left(\hat{R}^{t,s}_{t-1,s}(3,3)\right)^2}}\right) \\ \arctan 2\left(\frac{\hat{R}^{t,s}_{t-1,s}(3,2)}{\hat{R}^{t,s}_{t-1,s}(3,3)}\right) \end{bmatrix} = \begin{bmatrix} R^s_v\left(\delta^{t,v}_{t-1,v} - S^v_\circ\right) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.38}$$

The linearized measurement matrix then takes the form seen in Equation(3.39), where * simply denote the possibility of some non-zero value.

$$H = \frac{\partial}{\partial \mathbf{X}} G = \begin{bmatrix} 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.39}$$

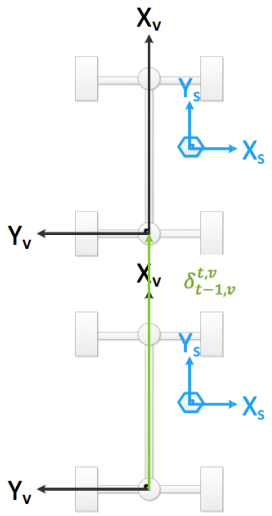

Figure 3.11: Straight Line Driving

Clearly, Equation(3.39) is rank deficient. While this is typically easy to maneuvers of non-constant attitude change in any real world test, some paths that could cause this to occur that may not be immediately obvious to the reader based on the aforementioned condition and are highlighted below. These situations of non-observability occur during straight line driving as in Figure 3.11, driving in perfect circles or purely rotational driving as seen in Figure 3.12, and holonomic driving or when sliding without rotation as seen in Figure 3.13. The reason for the lack of observablility for the case of constant rotation, can be extrapolated from both Appendix A and Equation 3.39, which is that $\hat{R}^{t,s}_{t-1,s}$ will be a constant value, and since a single measurement is insufficient repeating that same insufficient information will not achieve observability.

3.7 Conclusion

This chapter presented an auto-calibration technique based on the comparison of relative pose measurements from any two sensors capable of deriving these measurements. Additionally the outline of the implementation of this auto-calibration technique via an extended Kalman filter was presented. A number of different sensors where introduced and it was explained how these sensors could either determine their relative pose directly or how it

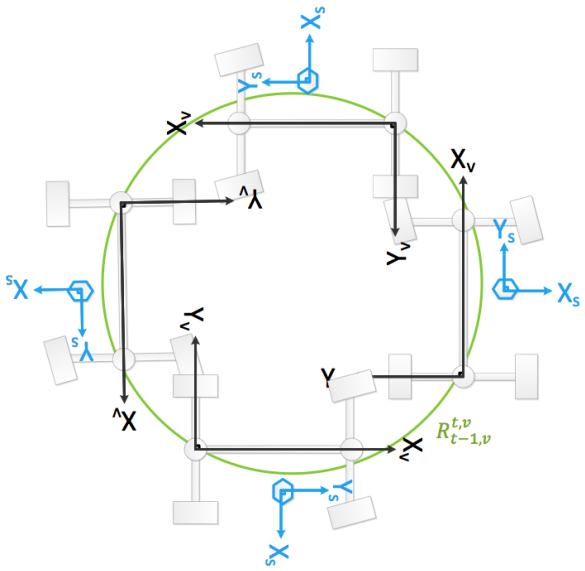

Figure 3.12: Concentric Circles

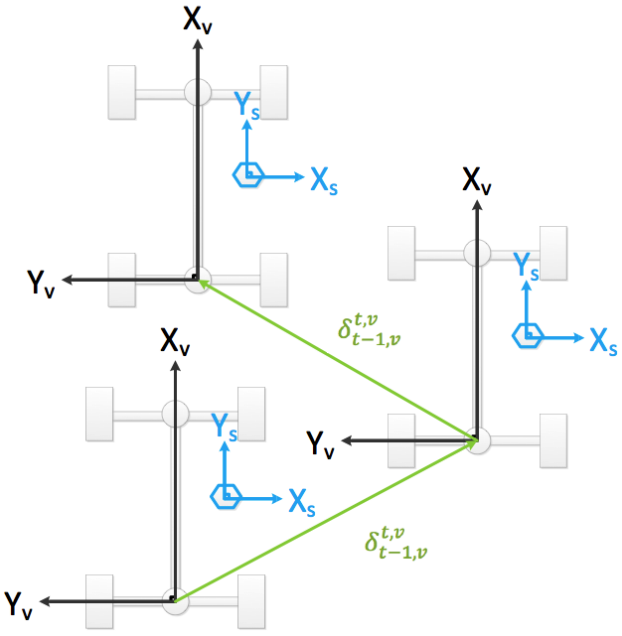

Figure 3.13: Holonomic Motion

could be inferred, as well as what parameters where desired when calibrating such a sensor. Finally, it was shown that this auto-calibration technique is observable provided that a non-degenerative path is taken. The observability analysis, while performed in three dimensions, can be easily extended to the two dimensional case, where the same principles apply.

Chapter 4

Validation of Auto-Calibration Technique

The auto-calibration technique outlined in Chapter 3, will be validated through the use of both simulated and real-world data, where a ground robot will move in three dimensional space while calibrating a three dimensional lidar to an INS using an EKF as outlined previously. Specifically the data used for the real-world analysis uses the publicly available data set from MIT taken during the DARPA Urban Challenge [31]. This dataset provides an extensive amount of information including multiple data sets of paths over 20km. Most importantly the data set includes calibration parameters as found by MIT during their laboratory calibration. During these tests the vehicle drove predominately at residential speeds and negotiated a number of obstacles all while having to contend with other vehicles. The two specific sensors that will be calibrated with this dataset are a Velodyne lidar and the Applanix INS system. The relative pose from each of these sensors will be obtained and used to find the translation and rotational offsets which comprise the mounting parameters. These mounting parameters will then be compared to those supplied with the dataset itself to ascertain the overall performance of the algorithm. Additional analysis will be performed in simulation, which will mimic the same route and dynamics as experienced by the vehicle in the actual data-set. Using the simulated data, an analysis of the sensitivity to the initial mounting error will be assessed. Additionally, the ability of the algorithm to detect and account for a change in the mounting parameters will be studied by blending a simulated mounting position for some length of time, and then proving the algorithm with real data. Note that the analysis conducted in this chapter analyzes performance over a large range of initial errors unlike the previous works of [42, 45, 98], where typically initial errors are on the order of 10cm or less and fractions of a degree.

4.1 Overview

The analysis performed in this chapter revolves around data taken from the DARPA Urban Challenge. The DARPA Urban Challenge was a multi-hour race where autonomous vehicles had to negotiate intersections as well as perform passing, merging, and parking maneuvers in an urban setting in the presence of both manned and unmanned vehicles representing a very realistic performance environment. The data provided by the MIT team includes both sensor data, time stamps, as well as the sensor mounting parameters. These parameters where found through a number of laboratory tests often involving static targets as outlined in [41].

The filter used to ascertain these mounting parameters will be the EKF detailed previously in Section 3.2. The states, and hence desired calibration parameters, are seen below as reproduced from Equation (3.6) which represent the desired position and attitude of the sensor to be calibrated.

$$\mathbf{X} = \begin{bmatrix} X_\circ & Y_\circ & Z_\circ & \psi & \theta & \phi \end{bmatrix}^{\mathrm{T}}$$

An overview of the sensors used for calibration is presented in Section 4.2 and the process of obtaining relative pose with them is outlined in Section 4.3. An analysis of the calibration technique in both simulation and using real data is shown in Sections 4.4 and 4.5.

4.2 Description of Sensors

The MIT vehicle as shown in Figure 4.1, is equipped with a 3D lidar, an INS, several push-broom style (2D) lidars as well as a Firefly MV Firewire camera system. The 3D lidar is specifically a Velodyne HDL-64e lidar and is mounted on a mast and is circled in Figure 4.1. It measures a full 360° sweep at 15 Hz using 64 lasers mounted on a spinning head that provide a vertical field of view of 26.8° and can be seen in detail in Figure 3.3. The data produced by this sensor is immense, and constitutes approximately one million measurements

Figure 4.1: Vehicle [19]

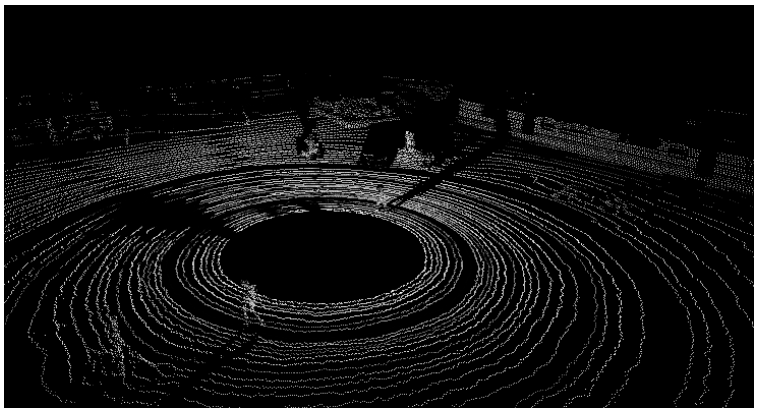

Figure 4.2: Raw Velodyne Scan [55]

every second and therefore can provide significant situational awareness of the environment as seen by a sample scan shown in Figure 4.2 [56]. The INS system is an Applanix POS-LV 220 GPS/INS which provides position, velocity, and orientation by combining differential GPS, a 1-degree of drift per hour IMU, and a 1024-count wheel encoder as shown previously Figure 3.7. It is worth noting that the raw IMU output from the INS was not provided. The INS system treats the center of the rear axle as its origin and all measurements are provided relative to this location. While the 2D lidars and camera system were not used specifically in the calibration process, the camera system was used to simply obtain a greater understanding of the surrounding environment the vehicle was operating within.

Table 4.1: True Mounting Parameters [31]

X-offset (m)	Y-offset (m)	Z-offset (m)	ψ (deg)	θ (deg)	ϕ (deg)
1.56	-0.004	2.55	91.03	-0.077	2.68

This dissertation treats the INS system as the reference frame denoted as v and the Velodyne from here on simply referenced as "lidar" as the sensor to be calibrated such that its coordinate frame is denoted as s. Note that because the INS origin is at the axle and the lidar is on a mast, there is a significant and non-trivial lever arm between these two sensors. Additionally as noted in Table 4.1 which lists the calibration values as found by the MIT team, there is also a significant rotational difference between the two sensors.

4.3 Obtaining Relative Pose

As noted in Section 3.1 the core of the calibration process relies on each sensor being capable of obtaining its relative pose regardless of the sensor type used. In Section 3.4 it was shown that obtaining relative pose from the INS is quite trivial and requires merely analyzing the difference in position and attitude from one time step to another. Obtaining the change in relative pose of the lidar is more challenging however because the lidar itself merely measures the distance from the laser emitter to environment and reports this distance in the coordinate frame of the lidar. It is therefore necessary to employ a method that is capable of determining the change in pose of the lidar from scan to scan using only these distance measurements. The specific method employed was that of iterative closest point (ICP) as introduced in Section 3.3.

A lidar scan will constitute one complete revolution of the lidar head. It is important to remember that during this process the vehicle remains in motion which will affect the perceived point cloud causing a smearing affect if left uncompensated. To account for vehicle motion the beginning of each lidar scan constitutes a new origin, and all successive points in the scan are transformed into this new origin by integrating the output of the INS. These successive compensated scans are then compared via the ICP algorithm which employed

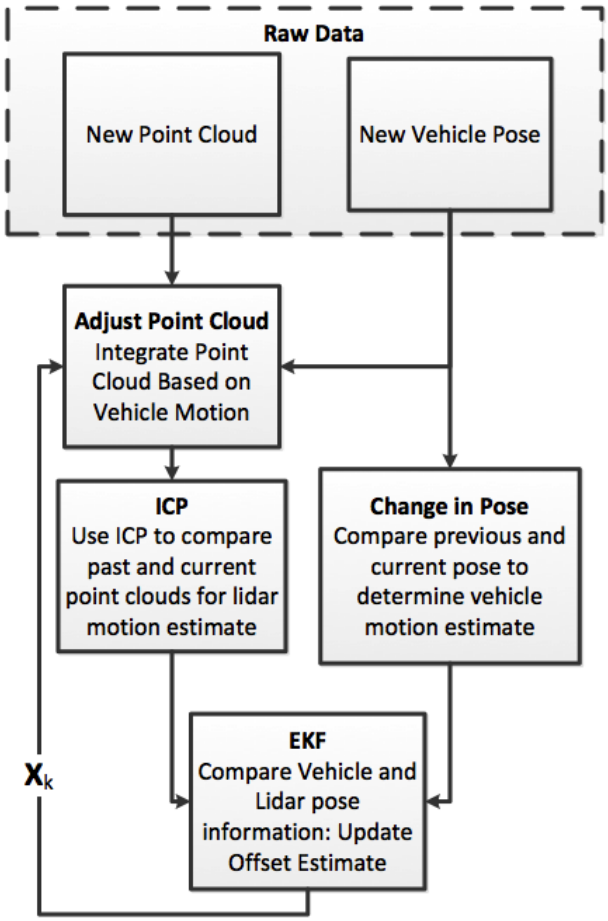

Figure 4.3: Process Flow

KD-tree matching with point-to-plane minimization where the worst 10% of the matches were rejected. The flow chart of the calibration process is shown in Figure 4.3. Note that the specific ICP algorithm used in this dissertation makes the inherent assumption that the environment is static, thus adding an additional assumption to the algorithm. However, if a more exotic ICP algorithm was employed such similar to [72] that could sufficiently function in dynamic environments, the constraint of having a static environment would obviously be unnecessary.

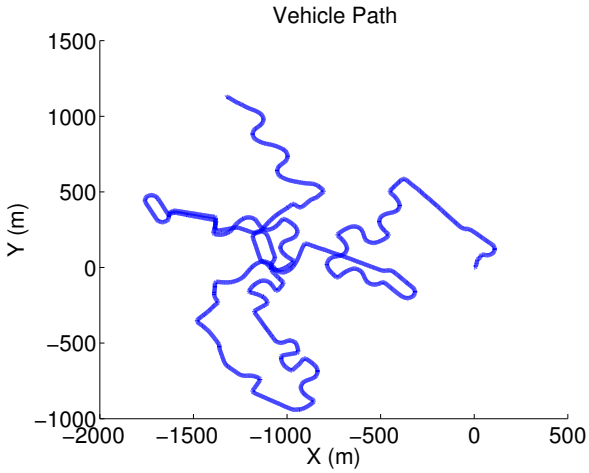

Figure 4.4: Simulated Vehicle Path

4.4 Simulation Evaluation

The vehicle was simulated over a portion of the Urban Grand Challenge route shown in Figure 4.4 which is effectively the path used from the experimental data used later. The path contains approximately 24,500 changes in pose. The INS was simulated by taking the INS from the real data set and treating it as truth, and changes in lidar pose were then calculated using the true mounting parameters from MIT as listed in Table 4.1. A Gaussian error of 5% of the distance traveled was introduced to all axes of the sensor data to simulate measurement errors. A Monte -Carlo simulation of 1,000 iterations was then performed where the initial conditions were allowed to vary in a range of $\mathbf{X} = [\pm 2.4m,\ \pm 0.96m\ \pm 1.78m,\ \pm 180°,\ \pm 90°,\ \pm 90°]^{\mathrm{T}}$. This initial condition range was selected because it is proportional to half the total vehicle width, length, and height with the lidar and mast, effectively testing performance for the lidar being mounted anywhere within the confines of the vehicle. The results for both translational error and rotational error of this simulated data run can be seen in Figures 4.5 and 4.6 for the EKF based approach. Additionally, Table 4.2 presents the numerical results from these simulations.

Note that the simulated data is capable of determining the true mounting parameters on average with a high degree of accuracy, with minimal error even with poor initial estimates.

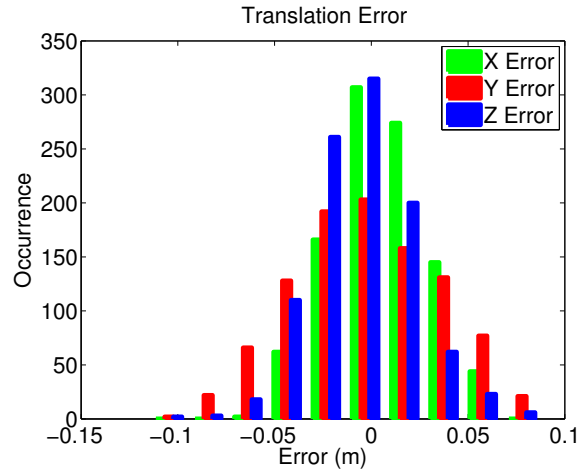

Figure 4.5: Translation Error Histogram for EKF Approach

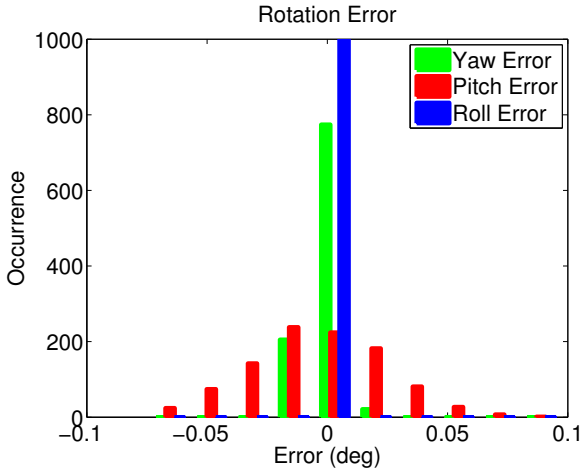

Figure 4.6: Rotation Error Histogram for EKF Approach

Table 4.2: Simulated Results

	EKF			
	Mean Final Value	Mean Error	Std	Max Error
X	1.5649 (m)	0.0049 (m)	0.0242 (m)	0.0659 (m)
Y	-0.0072 (m)	-0.0032 (m)	0.0371 (m)	0.0992 (m)
Z	2.5441 (m)	-0.0059 (m)	0.0259 (m)	0.1147 (m)
Yaw	91.03°	$-4.72° \times 10^{-4}$	0.0060°	0.0199°
Pitch	−0.08°	$-3.35° \times 10^{-4}$	0.0271°	0.0974°
Roll	2.68°	$-4.94° \times 10^{-4}$	0.0016°	0.0056°

Table 4.3: Simulated Results Varying Vehicle Motion

	0.1m & 0.1°		1m & 1°	
	Mean Error	Std	Mean Error	std
X	0.0022 (m)	0.0256 (m)	-0.0013 (m)	0.0292 (m)
Y	-0.0012 (m)	0.0242 (m)	0.0028 (m)	0.0233 (m)
Z	0.008 (m)	0.0314 (m)	-0.0047 (m)	0.0240 (m)
Yaw	$-0.0447°$	0.4889°	0.1036 (m)	0.7256°
Pitch	0.0483°	0.8339°	-0.0898 (m)	1.252°
Roll	0.1390°	1.0129°	0.0607 (m)	1.9334°

In an effort to better understand the causes of error, an additional series of Monte-Carlo simulation are performed where the amount of motion experienced by the vehicle is allowed to vary. Specifically, these simulations where performed not using the prior vehicle path so that greater control of sensor motion could be obtained. Each Monte-Carlo simulation consisted of 100 different iterations where the initial parameters were allowed to vary over the following range $\mathbf{X} = [\pm 2.4m,\ \pm 0.96m\ \pm 1.78m,\ \pm 180°,\ \pm 90°,\ \pm 90°]^{\mathrm{T}}$. The first Monte-Carlo simulation simulated vehicle motion as having an average change in motion 10cm, with an average change in rotation of 0.1° per measurement. The second Monte-Carlo Simulation increased this by an order of magnitude allowing the average change in motion to be 1m with an average change in rotation of 1° per measurement. These results also were simulated with a 15% error on ICP. The results of this analysis are presented in Table 4.3.

Despite the significant increase in the amount of motion the results do not show a meaningful gain in either average error or the standard deviation. In a further effort to determine the sensitivity of the algorithm additional analysis is performed by once again performing a series of Monte-Carlo simulations where the initial mounting parameters are allowed to vary over a range of $\mathbf{X} = [\pm 2.4m,\ \pm 0.96m\ \pm 1.78m,\ \pm 180°,\ \pm 90°,\ \pm 90°]^{\mathrm{T}}$. However, this time, the percent error will be varied. For these simulations the vehicle path shown in Figure 4.4 is used and the sensor measurements are generated using this vehicle path where the true mounting parameters are those provided in Table 4.1. For each percent

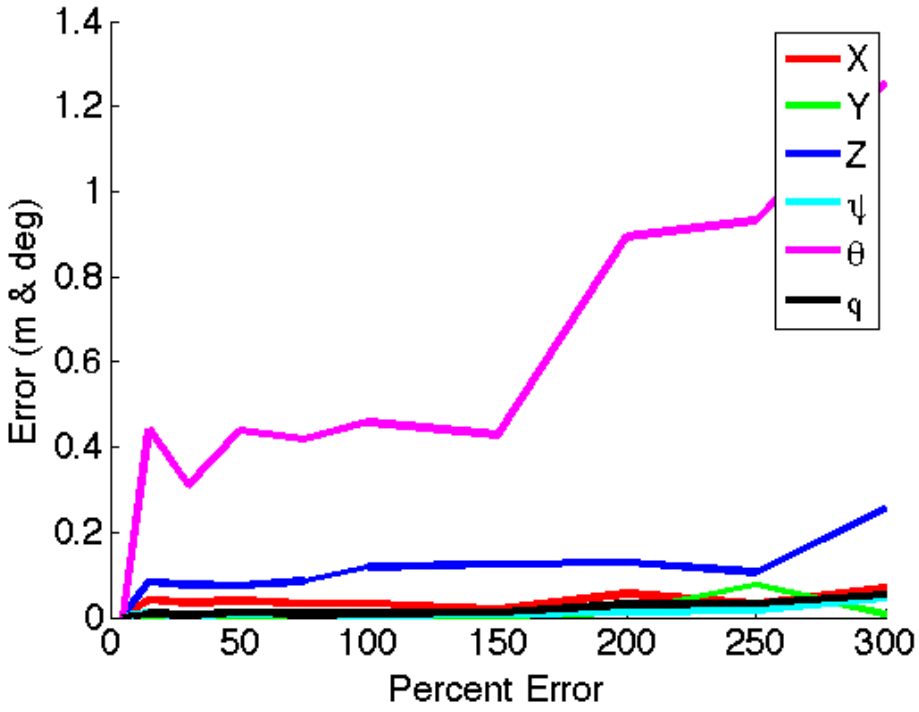

Figure 4.7: Average Error vs Varying Percent Error

Table 4.4: Average Motion Sensed in Sensor to be Calibrated

X	Y	Z	ψ	θ	ϕ
-0.0123 (m)	-0.4116 (m)	0.0044 (m)	$-0.0578°$	$0.001°$	$-0.0073°$

error presented, 100 unique simulations were performed. The absolute value of these results of the Monte-Carlo simulations are shown in Figure 4.7.

The trend from this plot is clear in that as the percent error increases, the average error also increases in the pitch and z translation results. However, the trend is not as apparent in the other states which hover near zero. The effect increasing percent error has on the standard deviation of these results however paints a clearer picture as seen in Figure 4.8.

The percent error error in the pitch and z translation states remains the highest. The actual amount of motion seen in each axis of the sensor to be calibrated is seen in Table 4.4.

Taking these results in context of the previous plots, it would appears that while the amount of noise on the ICP algorithm as simulated using percent error has a direct affect on the variation of the results, the disparity in motion between individual sensors can cause an

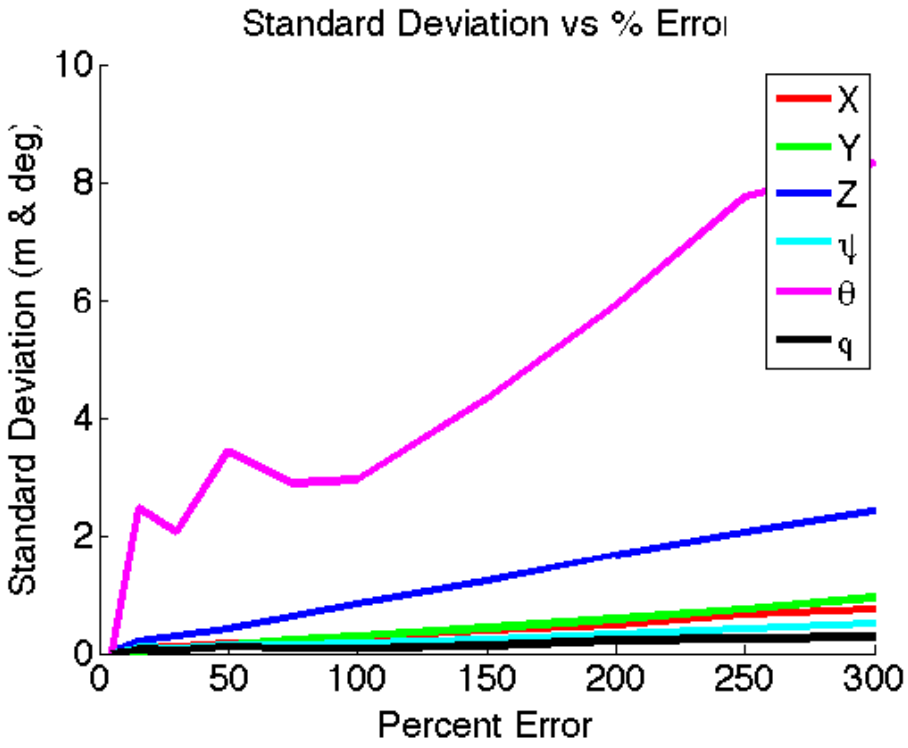

Figure 4.8: Standard Deviation vs Varying Percent Error

increase in the average error. However, this affect is minimized when all sensor experience roughly the same amount of excitation. Further evidence of this is provided through an additional series of Monte-Carlo simulations where the first test consisted of allowing the vehicle to experience an average motion of 1m and 1° in each axis with a 50% error injected into the simulated lidar ICP measurements. The second test consisted of using the average sensed motion values provided in Table 4.4 and injecting 15% error into both the vehicle and lidar ICP measurements. The third and final simulation once again used the sensed motion values shown in Table 4.4, but this time, 60% error was injected into both the vehicle and lidar ICP measurements. The results of these Monte-Carlo simulations are seen in Table 4.5, where again, the initial lidar mounting errors were allowed to vary over the range of vehicle ($\mathbf{X} = [\pm 2.4m,\ \pm 0.96m\ \pm 1.78m,\ \pm 180°,\ \pm 90°,\ \pm 90°]^{\mathrm{T}}$).

The simulated analysis indicates that having small perception errors and excitation in all axes provide the best results.

Table 4.5: Illustrating the Affects of Noise and Sensor Motion

	50% Error Uniform Motion	15% Error Small Varying Motion	60% Error Small Varying Motion
X	0.038 m	-0.007 m	0.150 m
Y	0.001 m	-0.023 m	0.551 m
Z	0.074 m	0.019 m	-0.22 m
θ	$-0.01°$	$-0.12°$	$-3.67°$
ϕ	$-0.44°$	$-0.27°$	$2.58°$
ψ	$-0.01°$	$-0.76°$	$6.93°$

Table 4.6: Full Comparison of Standardization to CRLB

	δ_x	δ_y	δ_z	ψ	θ	ϕ
Mean Difference	0.0520 m	0.0842 m	0.0864 m	$0.0014°$	$0.0044°$	$0.0049°$

4.4.1 Cramér–Rao Lower Bound (CRLB)

The CRLB is validated for the EKF using the vehicle path shown in Figure 4.9 which is comprised of 50 different changes in pose where the initial and every tenth change in pose are shown. The path length has been reduced from the path shown in Figure 4.4 so that trivially small standard deviations are not being compared. In the simulated path the vehicle experiences approximately a 35m change in its X axis, and approximately 50m of change in its Y and Z axes, along with approximately 45° of attitude change in all axes, thus providing significant excitation. For the CRLB analysis, sensor outputs were modeled using constant mounting parameters provided from the MIT dataset as seen in Table 4.1. A Gaussian noise of 10% of the distance traveled at each time-step was added to all change in pose measurements, and a total of 1,000 Monte Carlo simulations were conducted. The first 50 iterations of the Monte Carlo results are seen in Figure 4.10 where the CRLB and the calculated standard deviation are compared for the δ_x state. Additionally, Table 4.6 provides numerical results for the difference of all states, where the CRLB is compared to the calculated standard deviation. Because of the close agreement between the calculated standard deviations and their theoretical minima, it appears that this filter is both unbiased and efficient.

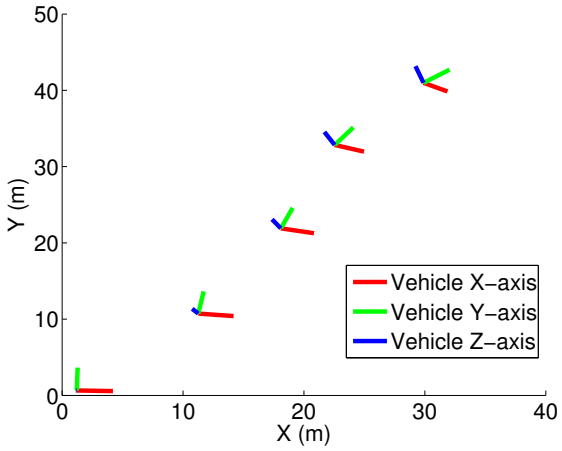

Figure 4.9: CRLB Vehicle Path

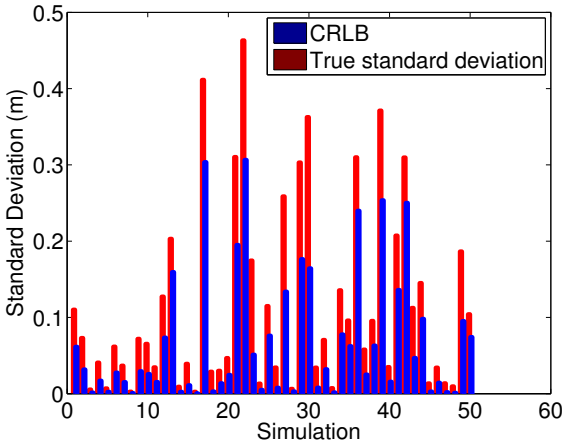

Figure 4.10: Calculated Standard Deviation Compared to CRLB of δ_x

4.4.2 Fault Detection

One of the benefits of the EKF approach is the ease at which fault detection and elimination schemes can be added. The benefit of fault detection is that should an erroneous measurement occur which could potentially worsen the calibration parameters, this can be detected, and not incorporated into the filter. Additionally, if the sensors actually shift due to environmental effects, this can be detected and compensated for automatically without input from the end user.

To detect a shift in the mounting parameters as well as reject erroneous measurements, a fault detection scheme is employed. The specific test is a χ^2 test on the innovation of the Kalman filter shown in Equation (4.1).

$$\left(\left(\begin{bmatrix} \delta_{t-1,s}^{t,s} \\ R_{t-1,s}^{t,s} \end{bmatrix} - \begin{bmatrix} \hat{\delta}_{t-1,s}^{t,s} \\ \hat{R}_{t-1,s}^{t,s} \end{bmatrix} \right) / \text{diag}\left(H_k P_k H_k^{\mathrm{T}} + R_k \right) \right)^2 < 3 \tag{4.1}$$

If this test fails, the measurement is rejected as an error. If this test fails three consecutive times, it is assumed to mean that the mounting location has been shifted. When it has been determined that a sensor has been shifted, the covariance matrix is reset to its original value and the current state is used to re-initialize the filter.

The process of detecting a mounting change is illustrated in Figure 4.11 and Tables 4.7, where the lidar sensor is modeled as changing during the auto-calibration process at iteration 2000. Because it was not possible to physically alter the mounting of the lidar during an actual run, this is done partially in simulation. This simulation was accomplished by creating the appropriate sensor outputs using the same sensor simulation setup outlined in Section 4.4 where a 5% error based on the distance traveled has been added to all measurements. At iteration 2000 the simulation data ends and is then switched for the real data from the MIT dataset, and is denoted by an orange circle in Figure 4.11. Note that once this change is made the filter reacts and begins converging to the true values. Figure 4.11 merely highlights the section of the fault detection and the actual processing of the MIT data set continued,

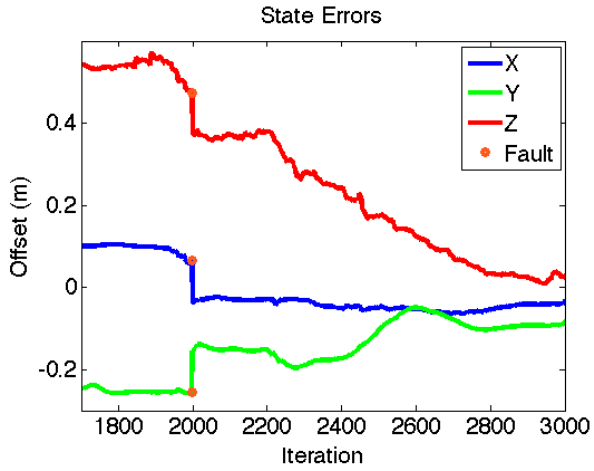

Figure 4.11: EKF Fault Detection

Table 4.7: EKF Fault Detection Final Error

δ_x	δ_y	δ_z	ψ	θ	ϕ
-0.0646 m	-0.0305 m	-0.1369 m	0.1622°	0.5531°	−0.1076°

but is not shown here for ease of viewing the desired area as the entire combined data set is approximately 26,500 iterations long.

The final error from this approach is seen in Table 4.7. Note that the erroneous data constituted approximately 7.5% of the total data, however, this erroneous portion had a noticeable impact on the results. Note that the majority of the error is in the δ_z translation and pitch state. The error in these states in unsurprising as they typically less excitation in comparison to the yaw and δ_x and δ_y states.

4.5 Experimental Validation with Truth

The specific dataset analyzed from MIT is the "MISSION2" log file which contains over two hours of data covering a 27km path. Due to the nature of the competition there are sections that contain several dynamic obstacles such as vehicles driving with or across the field of view of the lidar. Because the ICP algorithm assumes that the environment is static, some of these sections were removed from processing in which these types of dynamic obstacles seemed pervasive. The total vehicle path was subdivided into the used and rejected

sections of the path which are shown in Figure 4.12. This subdivision of data was first performed by splitting the entire dataset into two minute segments. Those segments that exhibited pervasive dynamic obstacles as ascertained via supplied time-tagged video along with additional post-processing were removed.

Additionally, a conservative threshold of 1/8m was chosen such that if the INS did not report a change in pose of at least this distance, no auto-calibration was attempted. The reason for this is two-fold. Firstly, as noted from the observability conditions the vehicle must move in order to determine the mounting parameter. Secondly this removes processing trivially small changes in pose which the ICP algorithm struggled with, often reported no change.

Because of the time burden associated with ICP combined with the length of the log file for the Monte Carlo simulations, ICP was performed in post-process and the same ICP results were used throughout the Monte-Carlo tests. This amounts to the "Adjust Point Cloud" block being removed in Figure 4.3. Then the calibration was performed for 1,000 different Monte Carlo runs where the initial conditions were drawn uniformly from the following range $\mathbf{X} = [\pm 2.4\text{m},\ \pm 0.96\text{m},\ \pm 1.78\text{m},\ \pm 180°,\ \pm 90°,\ \pm 90°]$. However, it was noted that the initial attitude estimate had the largest affect on overall performance. Therefore the Monte-Carlo analysis was repeated for varying ranges of initial attitude error. Shown in Table 4.8 are three different 1,000 run Monte-Carlo simulations in which the initial angular mounting error is varied while the initial translational error operates over the same range. From these results it can be deduced that when a relatively good initial estimate of the mounting parameters are available, the EKF method performs the quite well. These results indicate that the operator should know the alignment to within 90° of yaw and 45° of pitch and roll for best results. Similar results were obtained when the point clouds were compensated for motion with the "Adjust Point Cloud" block intact. However due to the significant time required for ICP, lengthy Monte-Carlo analysis was infeasible. A sample of these results can be seen in Table 4.9, where the initial mounting lidar is assumed to be directly on top of the INS,

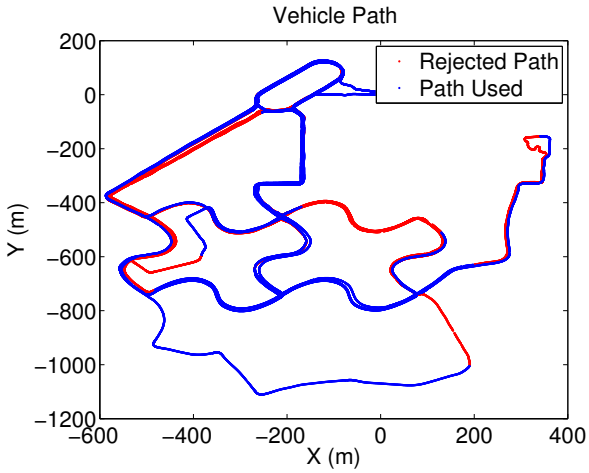

Figure 4.12: Vehicle Path Used

Table 4.8: Monte Carlo Results

	EKF		
Initial yaw error	±90°	±135°	±180°
Initial pitch error	±45°	±67.5°	±90°
Initial roll error	±45°	±67.5°	±90°
Final X error	0.0015m	-0.0412m	0.0401m
Final Y error	0.0003m	0.1372m	0.3197m
Final Z error	0.0326m	0.0531m	0.1556m
Final yaw error	$8.8° \times 10^{-6}$	1.8045°	5.1812°
Final pitch error	$-3.72° \times 10^{-4}$	−0.2337°	0.9259°
Final roll error	$1.5796° \times 10^{-5}$	0.5616°	0.8564°

hence $\mathbf{X} = [0\text{m},\ 0\text{m},\ 0\text{m},\ 0°,\ 0°,\ 0°]$. A plot of the EKF results for this specific run are shown in Figure 4.13, where the various state errors are shown to converge. Only the initial 3,000 iterations are shown of the approximately 24,500 total changes in pose.

The Monte-Carlo analysis demonstrates both that the simulation data of a 5% error in ICP greatly outperforms the actual data. Additionally it is of note that for larger initial mounting errors, the yaw is the largest rotation error, and y has the most translation error. Analyzing the ICP data, it is observed that on average, the motion in the lidar x-axis is roughly $40cm$ per measurement, while the y and z axes are 6mm and 1.5cm respectively. The yaw error being the lowest error for the 90° initial yaw error case is predictable given that it has an average motion of nearly 0.05° per measurement, while the pitch and roll axes

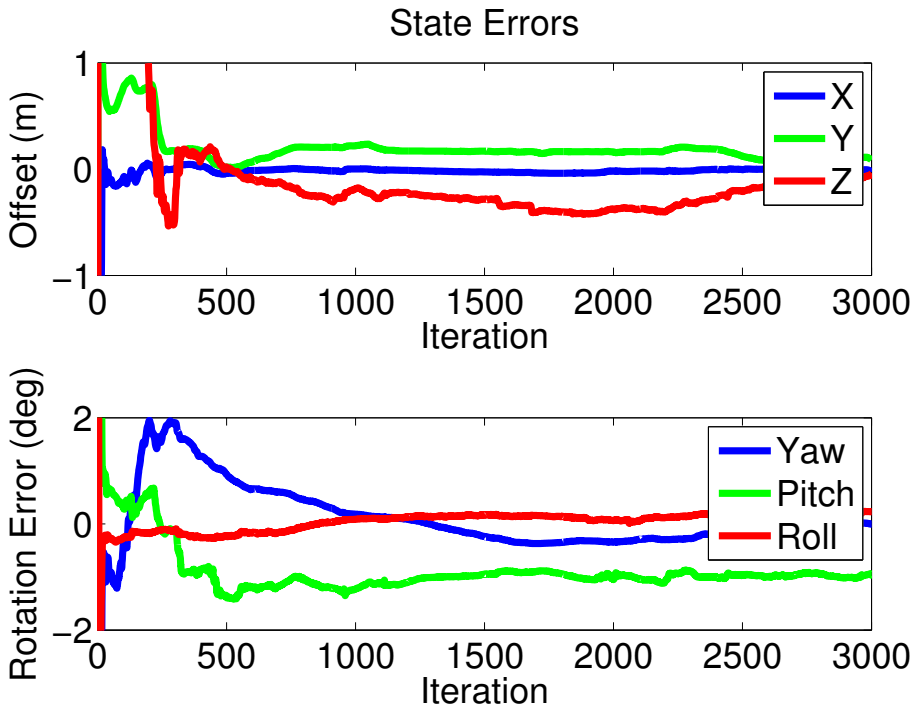

Figure 4.13: EKF State Errors

Table 4.9: Full Feedback EKF

	EKF	
	Final Value	Error
X	1.6116 m	0.0516 m
Y	0.0164 m	0.0205 m
Z	2.6285 m	0.0786 m
Yaw	91.0027°	−0.0229°
Pitch	−0.5457°	−0.4698°
Roll	2.6138°	−0.0688°

average 0.008° and 0.002° respectively. Additionally, using the simulated sensor values based on the same vehicle path as truth, and comparing those to the ICP values, it is observed that the ICP yaw has least least percent error at 3.6%, while the pitch and roll errors are 60% and 12.6%. Despite the reduced percent error, and increased motion, it appears that the large initial error cannot be fully overcome.

These results compare very favorable to other state of the art techniques used for lidar to IMU / INS calibration. For instance, the technique presented in [42] has a maximum angular error of 9° and a maximum offset of roughly 20cm, with results reported to be within sub degree and of roughly 2cm of accuracy. Similarly, in [45] initial errors are kept small with the largest initial error being roughly 2cm and again 9°. Note that results from [45] lacked truth and where merely an analysis of the mean and standard-deviation of the final mounting parameters. Additionally in [98], it is simply reported that errors are reduced to less than 5cm. Therefore, the technique developed in this dissertation seems to perform as well if not better than other similar techniques even when initialized with much greater errors.

4.6 Conclusion

This chapter demonstrated the ability to determine the three dimensional extrinsic calibration between two sensors, specifically an INS and Velodyne lidar to within cm level and sub degree accuracies. However, it was noted that the performance of the calibration tends to degrade in axes that see the least excitation and when the initial estimate of the mounting location differs more than 45° from the true mounting location. The ability of the developed algorithm to detect a change in mounting and compensate for it on the fly was also demonstrated. Due to the long calibration time associated with ICP on these large point clouds, the ability to implement this algorithm in real-time should be addressed by either using a more computationally efficient ICP algorithm or simply using fewer points.

Additionally it was shown that this technique compares favorably with other state of the art techniques even with large initial errors.

Chapter 5

SLACAM

Simultaneous localization auto-Calibration and mapping (SLACAM) is a technique that combines the functionality of both simultaneous localization and mapping (SLAM) with that of auto-calibration. The overall goal of SLACAM to be able to place a robot in an unknown environment with unknown sensor mounting locations, and have the robot explore while constructing a map of its surrounding, placing itself accurately within that map, all the while determining the mounting locations of the sensors on the robot. This is obviously quite challenging because typically SLAM assumes that the mounting locations of the sensors are fixed. This is because the sensors changing mounting locations on the robot has the ability to effectively shift walls or the robots location within the map. Additionally auto-calibration techniques often assume known environmental properties or accurate histories of motion, which are not always available when operating in unknown environments. This chapter will develop the SLACAM technique such that a two-dimensional ground robot will be capable of navigating in an indoor environment while using only an IMU, lidar, and wheel odometry measurements whilst calibrating these sensors. Note that this work expands on previous techniques such as [49, 37, 34][49, 37, 34] where only two sensors are calibrated, and whose filter lack the robustness to use the same auto-calibration technique for a wide variety of sensors.

5.1 Overview

The SLACAM technique presented will marry EKF based SLAM with the EKF based extrinsic auto-calibration technique based on relative pose estimates first introduced in Chapter 3, and validated in Chapter 4. Note that a monolithic EKF structure is used which is

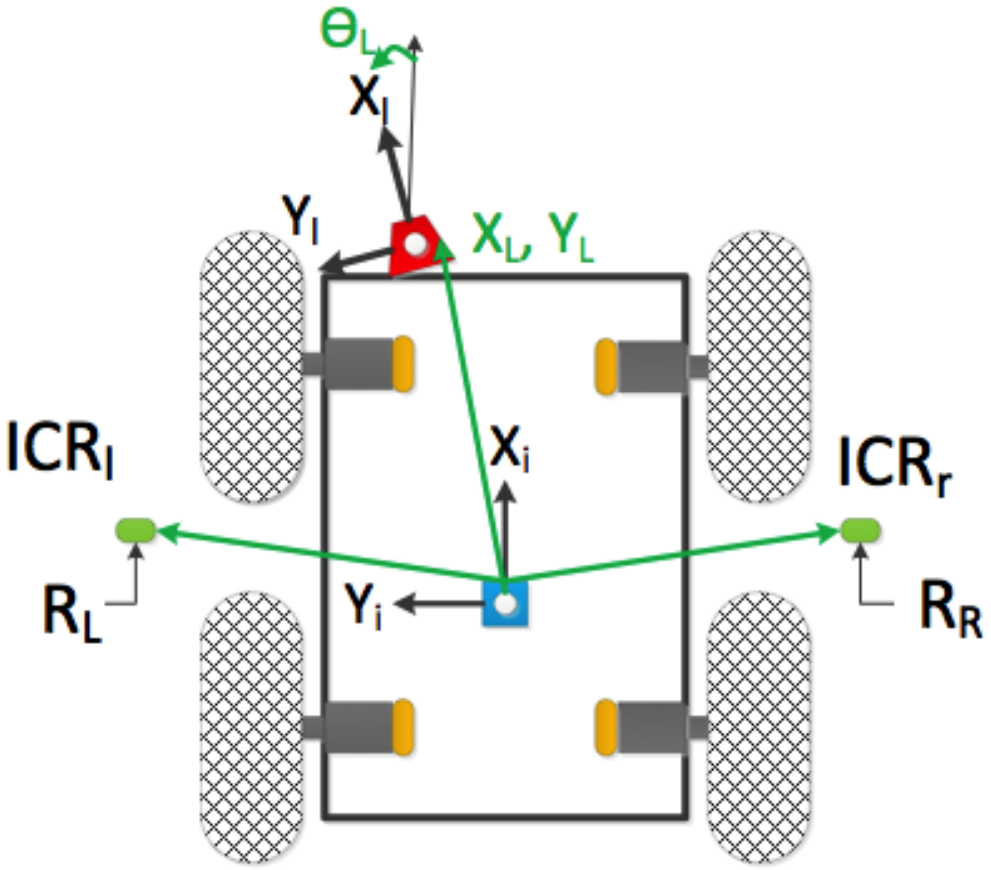

Figure 5.1: Desired Calibration Values

theorized to provide optimal results over a series of federated filters as seen in [27]. Specifically, this filter will attempt to generate a map which the robot is capable of using for navigation, as well as locate the robot accurately within the map, while determining the necessary values to calibrate a differential drive odometry system and lidar to and IMU, where the desired and unknown calibration values are outlined in Figure 5.1.

A flow-chart of the technique is seen in Figure 5.2, which provides a high level outline of the process and the steps that must be accomplished. Note that while this flow chart specifically refers to odometry and lidar measurements, sensors with enough perception information to be used with SLAM could be substituted for the lidar, and any additional aiding sensor could be used for odometry with no inherent alterations to either the SLAM or auto-calibration processes.

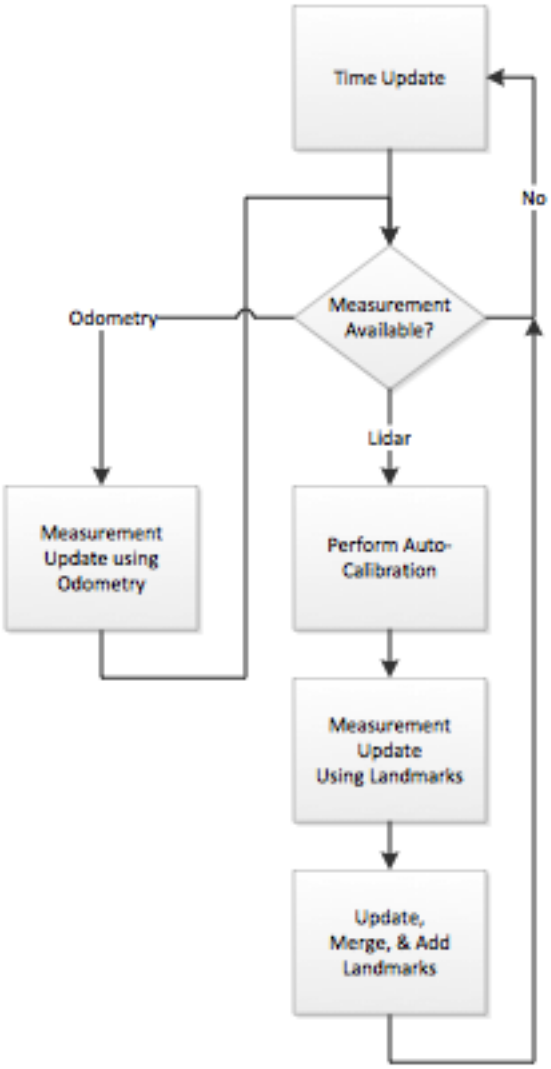

Figure 5.2: SLACAM Process

The process presented in this dissertation revolves around an IMU whose data is processed to determine velocity as well as changes in position and heading. As odometry becomes available, this sensor will aid in updating both the position and velocity of the robot. Finally as lidar becomes available, the auto-calibration technique will be performed followed by a measurement update using the landmarks extracted by the lidar which will update both the landmarks in the map, the position of the robot in the map, and potentially provide some aiding to the calibration measurements. Finally new landmarks will be merged with existing landmarks or added outright, and the process will continue. This technique is developed in greater detail in Section 5.2. An overview of the various sensors measurements are provided in Section 5.3 with detailed landmark information in Section 5.3.3.2 through 5.3.3.5. An overview of the calibration procedure is provided in Sections 5.4 and 5.5. Finally the observability of this technique is provided in Section 5.7.

5.2 Technique Development

This Section will be develop the general outline of the the filter used and the states to be estimated. Specifically, this filter is developed for a skid-steer robot operating with three degrees of freedom in a structured environment. SLACAM will be formulated in terms of an extended Kalman filter (EKF) whose generic time update can be seen in Equation (5.1), and generic measurement update can be seen in Equation (5.2).

$$\begin{aligned} \hat{\mathbf{X}}_k^- &= F\left(\hat{\mathbf{X}}_{k-1}^-,\ u_k\right) \\ A &= \tfrac{\partial F(X_{k-1},\ u_k)}{\partial X_k} \\ P_k^- &= AP_{k-1}A^{\mathrm{T}} + Q_{k-1} \end{aligned} \tag{5.1}$$

$$
\begin{aligned}
H_k &= \tfrac{\partial}{\partial X_k} G\left(X_k\right) \\
K_k &= P_k^- H_k^{\mathrm{T}} \left(H_k P_k^- H_k^{\mathrm{T}} + R_k\right)^{-1} \\
y_k &= Z_k - G\left(\hat{\mathbf{X}}_k\right) \\
\hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_k^- + K_k y_k \\
P_k &= \left(I - K_k H_k\right) P_k^-
\end{aligned}
\tag{5.2}
$$

Note that $\hat{\mathbf{X}}_k$ represents the state vector comprising the variables that are sought to be estimated, and that F represents the non-linear state propagation which will revolve around the IMU. Additionally P_k represents the covariance matrix which will become critical in the SLACAM process. Additionally note that the non-linear measurement equation is denoted as G. The desired states which comprise $\hat{\mathbf{X}}_k$ can be found in Equation (5.3).

$$
\hat{\mathbf{X}}_k = \begin{bmatrix}
\begin{bmatrix} X & Y & \Theta & \dot{X} & \dot{Y} & b_x & b_y & b_z \end{bmatrix}^{\mathrm{T}} \\
\begin{bmatrix} X_L & Y_L & \theta_L \end{bmatrix}^{\mathrm{T}} \\
\begin{bmatrix} X_{ICR} & Y_{ICR_L} & Y_{ICR_R} & \zeta_L & \zeta_R \end{bmatrix}^{\mathrm{T}} \\
\begin{bmatrix} L_{\rho_1} & L_{\alpha_1} & \cdots & L_{\rho_n} & L_{\alpha_n} \end{bmatrix}^{\mathrm{T}}
\end{bmatrix}
\tag{5.3}
$$

The first row of the states provided in Equation (5.3) represent the location of the robot in the navigation frame ($X,\ Y$) as well as the heading within that frame Θ, followed by the velocity of the robot ($\dot{X}\ \dot{Y}$), and finally inertial biases related to the accelerometer and gyro ($b_x,\ b_y,\ b_z$) which will be expanded on in greater detail in Section 5.3.1. The second row of Equation (5.3) represents the desired lidar calibration parameters which are the lateral and angular offset from the IMU which is the three degree-of-freedom equivalent to the calibration parameters discussed in Chapter 3. 'The third row of Equation (5.3) represents parameters necessary to calibrate the odometry on a skid-steer vehicle, which are the instantaneous centers of rotation for the left and right sides ($X_{ICR},\ Y_{ICR_L},\ Y_{ICR_R}$), and the effective wheel radii ($\zeta_L,\ \zeta_R$), as outlined in Section 3.5.

The final row of Equation (5.3) represents the landmarks, which will be represented in polar coordinates as having some range ρ and angle α. Hence, each landmark will contain both of these parameters, and the state matrix will be capable of growing to accommodate up to n landmarks. Greater details on the acquisition, addition, and merging of landmarks will appear in Section 5.3.3. Note that P_k is the covariance matrix that affectively acts as a measure of confidence or uncertainty of the variables that are being estimated. Note that the diagonal elements as seen in Equation (5.4) represent the variance of the variable, and the off-diagonal elements represent the correlation between the two elements in question.

$$P_k = \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 & \sigma_{1,\cdots}^2 & \sigma_{1,n}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 & \sigma_{2,\cdots}^2 & \sigma_{2,n}^2 \\ \sigma_{\cdots,1}^2 & \sigma_{\cdots,2}^2 & \ddots & \vdots \\ \sigma_{n,1}^2 & \sigma_{n,2}^2 & \cdots & \sigma_{n,n}^2 \end{bmatrix}_k \tag{5.4}$$

Thus accurately describing the correlation between landmarks and sensors is necessary for optimal performance. Note the covariance for a specific element can be defined using Equation (5.5), where $\rho_{x,y}$ represents the correlation coefficient.

$$\sigma_{x,y}^2 = \operatorname{cov}(x, y) = E\left[(x - \mu_x)(y - \mu_y)\right] = \rho_{x,y}\sigma_x\sigma_y \tag{5.5}$$

The equations for both the time and measurement updates are formulated in Section 5.3.

5.3 Measurements

The platform used to test and validate SLACAM is the iRobot ATRV which is seen in Figure 5.3. Note that the robot presented has been altered from its stock form to include an updated IMU , a new Lidar, and new odometry sensors. While a GPS antenna is present in the photo on the robot, it was not used in this research. The following sub-sections will discuss the measurements from the various sensors, and how they are incorporated into the filter. Specifically, the IMU will be discussed in 5.3.1, which will act as the input to the

Figure 5.3: iRobot ATRV Test Platform

time update portion of the EKF. This will be followed by odometry information in 5.3.2 and lidar information in 5.3.3 which both are incorporated into the measurement update. In addition to the specific measurements made by the lidar, the extraction of landmarks and their incorporation into the filter is discussed.

5.3.1 IMU

The IMU provides measurements of specific fore and angular rate which will be used for both navigation and calibration purposes. The outputs of the IMU will be integrated to determine the velocity and change in position of the robot. The IMU used is a MEMS grade Crossbow 440 which will provide measurements at 100Hz, show in Figure 5.4 .

Note that an IMU is subject to various measurement errors, and this especially true for MEMS IMUs which can only be used unaided for brief periods of time. Therefore, to aid the navigation solution, the major errors inherent in the IMU will be modeled and estimated. As noted in [26] the major sources of error for MEMS IMUs are a Gaussian noise (w), and

Figure 5.4: IMU Crossbow 440

a bias (b) for both the accels and gyros, such that the outputs of the IMU can be modeled as seen in Equation (5.6).

$$\begin{aligned} \widetilde{A}_x &= A_x + b_x + w_x \\ \widetilde{A}_y &= A_y + b_y + w_y \\ \widetilde{G}_z &= G_z + b_z + w_z \end{aligned} \tag{5.6}$$

Note that $\widetilde{A}_x$, $\widetilde{A}_y$, and $\widetilde{G}_z$ represented the corrupted measurements for the accels and gyros in the x, y, and z axes, while A_x, A_y, and G_z represented the true acceleration and rotation rate experienced by the robot. The white noise errors are modeled as seen in Equation (5.7) as seen in [26].

$$
\begin{aligned}
w_x &= N\left(0, \sigma_{A_x}^2 dt\right) \\
w_y &= N\left(0, \sigma_{A_y}^2 dt\right) \\
w_z &= N\left(0, \sigma_{G_z}^2 dt\right)
\end{aligned} \tag{5.7}
$$

The bias are modeled as first order Markov processes as defined in Equation (5.8).

$$
\begin{aligned}
b_x &\subset \left[\ 0, \quad \sigma_{A_{x_{bias}}}^2\ \right], \quad \dot{b}_x = -\tfrac{1}{\tau_{A_x}} b_x + w_x \\
b_y &\subset \left[\ 0, \quad \sigma_{A_{y_{bias}}}^2\ \right], \quad \dot{b}_y = -\tfrac{1}{\tau_{A_y}} b_y + w_y \\
b_z &\subset \left[\ 0, \quad \sigma_{G_{z_{bias}}}^2\ \right], \quad \dot{b}_z = -\tfrac{1}{\tau_{G_z}} b_z + w_z
\end{aligned} \tag{5.8}
$$

The sensor model now exists to account for the IMU errors such that the measurements can be accurately incorporated into the filter. However, the navigation frame of the robot must be accounted for first. The robot will operate in a two-dimensional frame, where both the body frame and the navigation frame will be defined as Z-up frames. Thus the frames can be aligned by merely a rotation about the Z axis by some amount Θ, as seen in Equation (5.9).

$$
C_b^n = \left(C_n^b\right)^{\mathrm{T}} = R_z^{\mathrm{T}}(\Theta) = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{bmatrix} \tag{5.9}
$$

The IMU measurements as defined in Equation (5.10) become an input to the state propagation outlined previously in Equation (5.1), where $F\left(\hat{\mathbf{X}}_{k-1}^{-},\ u_k\right)$ is now defined in continuous time in Equation (5.11).

$$
u(t) = \begin{bmatrix} \widetilde{A}_x(t) & \widetilde{A}_y(t) & \widetilde{G}_z(t) \end{bmatrix}^{\mathrm{T}} \tag{5.10}
$$

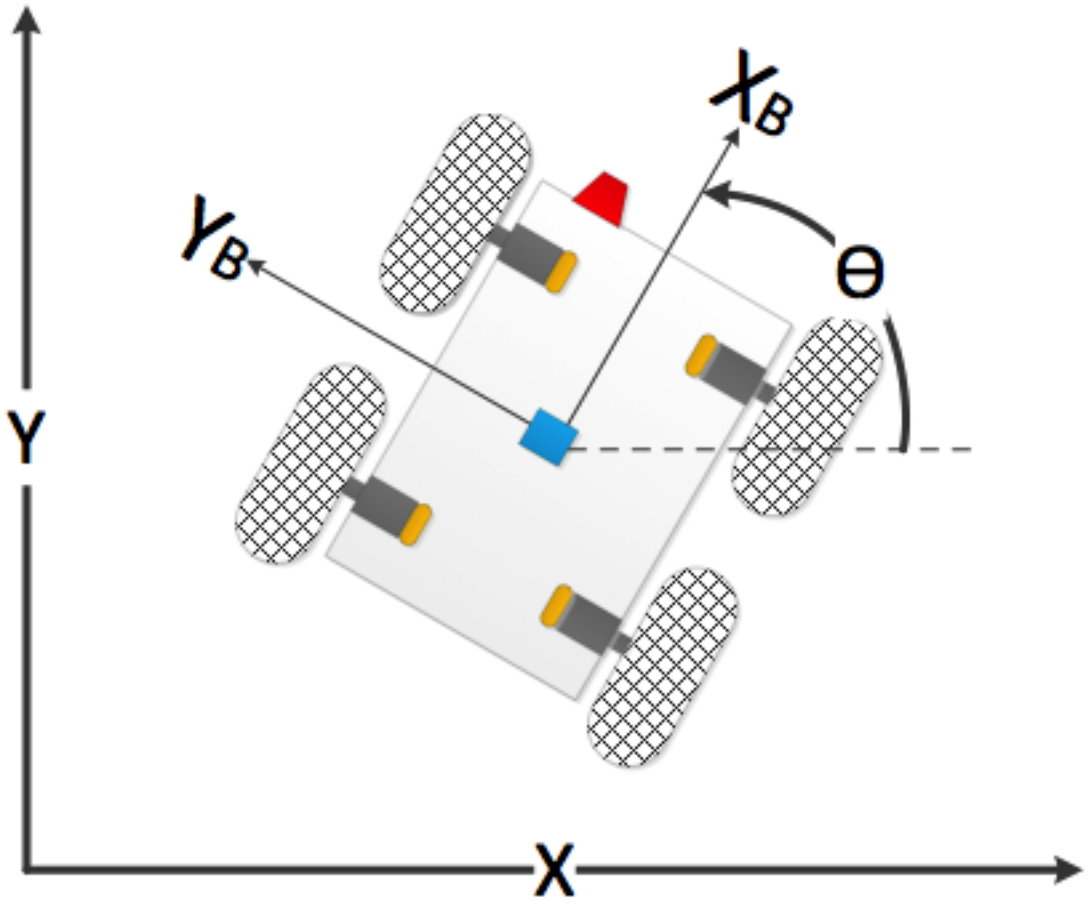

Figure 5.5: IMU Body to Nav Frame

$$\dot{\hat{\mathbf{X}}}(t) = \begin{bmatrix} \dot{X}(t) \\ \dot{Y}(t) \\ \dot{\Theta}(t) \\ --- \\ \ddot{X}(t) \\ \ddot{Y}(t) \\ --- \\ \dot{b}_x(t) \\ \dot{b}_y(t) \\ \dot{b}_z(t) \\ \dot{X}_L \\ \vdots \\ \dot{L}\alpha_n \end{bmatrix}, \; F\left(\hat{\mathbf{X}},\; \mu\right) = \begin{bmatrix} \dot{X}(t-1) \\ \dot{Y}(t-1) \\ \widetilde{G}_z(t) - b_z(t-1) \\ --- \\ C_b^n \begin{bmatrix} \widetilde{A}_x(t) - b_x(t-1) \\ \widetilde{A}_y(t) - b_y(t-1) \end{bmatrix} \\ --- \\ -\frac{1}{\tau_{A_x}} b_x \\ -\frac{1}{\tau_{A_y}} b_y \\ -\frac{1}{\tau_{G_z}} b_z \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.11}$$

Note that the calibration parameters along with the landmark locations will be modeled as being constant, and hence will not change during the time update. In order to propagate the states and covariance forward in time, fourth order Runge-Kutta is used as defined in Equation (5.12), where K and $K_p$are defined in Equations (5.13) and (5.14) where dt is the sample rate of the IMU.

$$\begin{aligned} \hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_{k-1} + \tfrac{1}{6}K_1 + \tfrac{1}{3}K_2 + \tfrac{1}{3}K_3 + \tfrac{1}{6}K_4 \\ \hat{P}_k &= \hat{P}_{k-1} + \tfrac{1}{6}K_{p1} + \tfrac{1}{3}K_{p2} + \tfrac{1}{3}K_{p3} + \tfrac{1}{6}K_{p4} \end{aligned} \tag{5.12}$$

$$\begin{aligned} K_1 &= F\left(\hat{\mathbf{X}}_{k-1},\; \mu\right) dt \\ K_2 &= F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_1}{2},\; \mu\right) dt \\ K_3 &= F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_2}{2},\; \mu\right) dt \\ K_4 &= F\left(\hat{\mathbf{X}}_{k-1} + K_3,\; \mu\right) dt \end{aligned} \tag{5.13}$$

$$
\begin{aligned}
K_{p1} &= \left(A_1 P_{k-1} + P_{k-1} A_1^{\mathrm{T}} + Q\right) dt \\
K_{p2} &= \left(A_2 P_{k-1} \tfrac{K_{p1}}{2} + P_{k-1} A_2^{\mathrm{T}} \tfrac{K_{p1}}{2} + Q\right) dt \\
K_{p3} &= \left(A_3 P_{k-1} \tfrac{K_{p2}}{2} + P_{k-1} A_3^{\mathrm{T}} \tfrac{K_{p2}}{2} + Q\right) dt \\
K_{p4} &= \left(A_4 P_{k-1} K_{p3} + P_{k-1} A_4^{\mathrm{T}} K_{p3} + Q\right) dt
\end{aligned}
\tag{5.14}
$$

Where:

$$
\begin{aligned}
A_1 &= \tfrac{\partial}{\partial \hat{\mathbf{X}}_{k-1}} F\left(\hat{\mathbf{X}}_{k-1},\ \mu\right) \\
A_2 &= \tfrac{\partial}{\partial \hat{\mathbf{X}}_{k-1}} F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_1}{2},\ \mu\right) \\
A_3 &= \tfrac{\partial}{\partial \hat{\mathbf{X}}_{k-1}} F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_2}{2},\ \mu\right) \\
A_4 &= \tfrac{\partial}{\partial \hat{\mathbf{X}}_{k-1}} F\left(\hat{\mathbf{X}}_{k-1} + K_3,\ \mu\right)
\end{aligned}
\tag{5.15}
$$

and Q is the process noise defined in Equation (5.16).

$$
Q = \operatorname{diag}\left[\ \sigma_{A_x}^2 dt^2,\ \ \sigma_{A_y}^2 dt^2,\ \ \sigma_{G_z}^2 dt,\ \ \sigma_{A_x}^2 dt,\ \ \sigma_{A_y}^2 dt,\ \ \sigma_{G_z}^2 dt,\ \ \sigma_{A_{x_{bias}}}^2 \quad \sigma_{A_{y_{bias}}}^2 \quad \sigma_{G_{z_{bias}}}^2 \quad \epsilon \cdots \epsilon\ \right]
\tag{5.16}
$$

Note that the calibration parameters and landmarks are modeled as having some very small process noise for the ability to tune the filter. By virtue of starting in an unknown location, the user can define an initial location, and covariance on this location. While typically, a location of $(0, 0)$ is selected as the initial location, this is not necessary, and is often selected as a matter of connivence. However, because the initial location is user defined, and the assumption is made that the robot is static when it is initialized, the initial covariance on both the position and velocity is set to zero.

5.3.2 Odometry

The odometry sensors are wheel encoders attached to the shaft of each of the four drive motors as seen in Figure 5.6, where the wheel encoders are highlighted in orange. However,

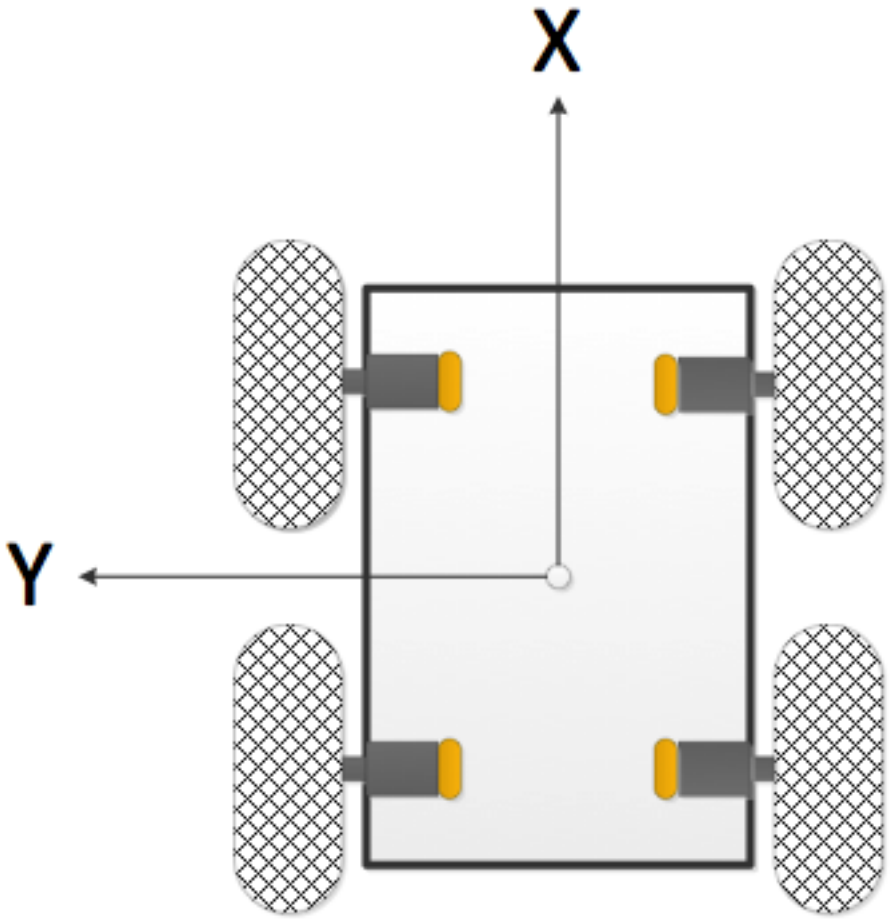

Figure 5.6: Wheel Encoders (shown in orange)

there are a series of gear reductions between the motor shaft and the actual wheel such that for each rotation of the tire, the encoder rotates 44,400 times, providing a high resolution measurement of how far the wheel has turned. From these measurements, the change in robot pose will be calculated in Section 5.3.2.1. Additionally, these sensors have the added benefit of providing feedback for when the robot is not moving, and thus provides a unique opportunity to update IMU biases which will be presented in Section 5.3.2.2.

5.3.2.1 Odometry Motion and Measurements

Recall from Section 3.5 that the odometry relies on estimating motion using instantaneous centers of rotation (ICR) as seen in Figure 5.7. Additionally recall that the velocity

of the robot in both the X and Y directions as well as the rotation rate can be calculated using Equation (5.17) which is also the measurement update in the EKF for the odometry when the robot is determined to be moving.

$$Z_{k,\text{odom}} = \begin{bmatrix} v_x^b \\ v_y^b \\ \omega_z \end{bmatrix} = \frac{1}{(y_{ICR_l} - y_{ICR_r})} \begin{bmatrix} -R_l y_{ICR_r} & R_r y_{ICR_l} \\ -R_l x_{ICR} & R_r x_{ICR} \\ -R_l & R_r \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{5.17}$$

However, the wheel encoders provide only a change in ticks which is incremented when the wheels are rotated forwards, and decrements when the wheels are rotated in reverse, thus these rotation rates must be calculated, as seen Equation (5.18).

$$\omega_l = \omega_r = \frac{Odom(t) - Odom(t-1)}{tpr} \frac{2\pi}{dt} \tag{5.18}$$

Note that *tpr* as noted in Section 3.5, denotes the ticks per revolution which for this sensor would be $44,400$. Additionally, the dt for the odometry system would be 0.02s (50Hz).

Note that the values used to navigate the robot are also the values that are to be estimated. Because of this, the values of the wheel radii and ICR are chosen such that the wheel radii are near the measured estimate of the true wheel radius, and the x_{ICR} is initialized to zero, and $y_{ICR,l}$ and y_{ICR_r} are such that they lay between the two wheels, effectively initializing the robot to a differential drive system. This however will quickly be corrected as the true values are determined. Also note, that if the *tpr* is incorrect, the wheel radii simply act as a scale factor, and will automatically compensate for any errors in estimating gear ratios and the like. Because the robot wheels have a tendency to slip when the motors are first engaged, the odometry measurements are not used until ten consecutive non-stationary measurements are observed. Thus, when $Z_{k,odom}$ is available, the corresponding prediction of the body frame velocity and rotation rate is calculated in Equation (5.19), where $C_b^n = R_z(\Theta)$ is same as that shown previously in Equation (5.9).

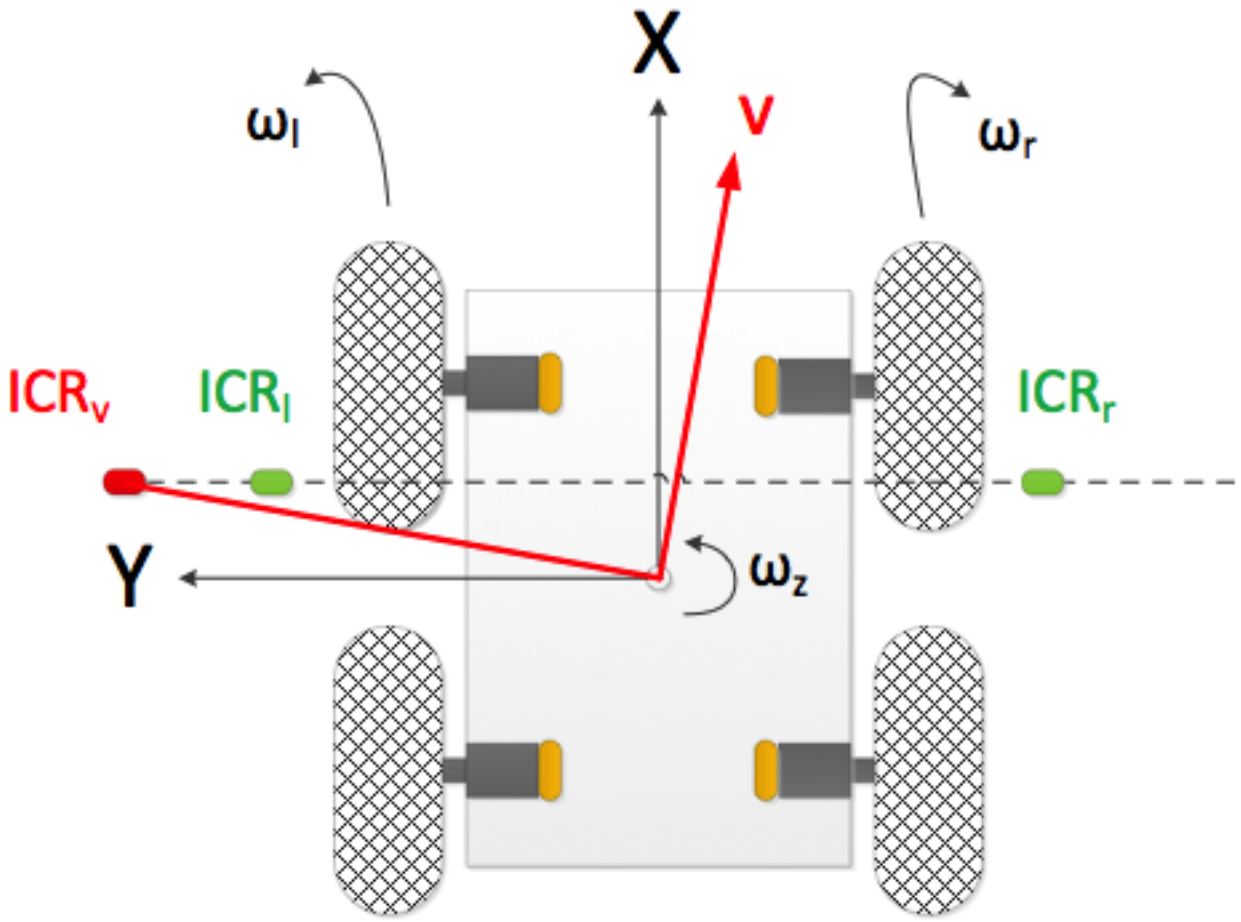

Figure 5.7: Desired Odometry Values

$$G_{k,\text{odom}} = \begin{bmatrix} \hat{v}_x^b \\ \hat{v}_y^b \\ \hat{\omega}_z \end{bmatrix} = \begin{bmatrix} R_z^{\mathrm{T}}\left(\hat{\mathbf{X}}_\mathbf{k}(3)\right) \begin{bmatrix} \hat{\mathbf{X}}_k(4) \\ \hat{\mathbf{X}}_k(5) \end{bmatrix} \\ - - - \\ \widetilde{G}_z(t) - \hat{\mathbf{X}}_k(8) \end{bmatrix} = \begin{bmatrix} R_z^{\mathrm{T}}(\Theta) \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \\ - - - \\ \widetilde{G}_z(t) - b_z \end{bmatrix} \tag{5.19}$$

5.3.2.2 Zero-Velocity Update (ZUPT)

The odometry system, in addition to aiding the estimate of robot pose, has the unique ability to ascertain when the robot is static. This is very beneficial, because the calibration algorithm relies on change of motion, and hence this provides an additional method to determine when not to attempt to calibrate, as well as the ability to update the IMU biases. In similar fashion to determining when the robot is moving, the odometry system does not treat the robot as static until there have been 125 consecutive measurements of no change in odometry. This is important, because the robot has the ability to slide slightly when coming to and abrupt halt, and this eliminates these brief errors from affecting the filter. Specifically, this type of static update is known as a zero-velocity update (ZUPT). Thus, when the ZUPT is performed, the IMU biases can be directly measured as the IMU output as seen in Equation (5.20).

$$Z_{k,\text{zupt}} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \widetilde{A}_x(t) \\ \widetilde{A}_y(t) \\ \widetilde{G}_z(t) \end{bmatrix} \tag{5.20}$$

Naturally, the predicted bias values will be the states themselves as seen in Equation (5.21).

$$G_{k,\text{zupt}} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{X}}_k(6) \\ \hat{\mathbf{X}}_k(7) \\ \hat{\mathbf{X}}_k(8) \end{bmatrix} \tag{5.21}$$

5.3.3 Lidar

The lidar is the perception sensor which the SLACAM algorithm revolves around. The lidar is capable of taking in a wealth of information which will be used to determine the change in pose of the lidar itself, as well as the navigable landmarks. It is through these landmarks extracted by the lidar that the robot will build a map of the environment, as well as use them to navigate. Note that the lidar could be replaced by stereo-vision or some other information rich perception sensor capable of extracting landmarks with a bounded drift. An overview of the actual lidar and its measurements will be outlined in Section 5.3.3.1, and the process of extracting landmarks, adding them to the state matrix, and merging new landmarks with existing landmarks will be covered in Sections 5.3.3.2 through 5.3.3.5.

5.3.3.1 Overview of Lidar used and Measurements

A lidar is an active sensor such that it emits a pulse of laser light and measures a returning pulse. Based on the difference in time between the emission and reflection, the distance between the lidar and the object reflecting the light can be determined. While lidars use different mechanisms for actuating the light emitter and detector, the SICK LMS151 used in this dissertation and seen in Figure 5.8 actually rotates a mirror which directs the laser in a horizontal sweep. Through this actuation, the lidar has a 270° field of view (FOV), and has a scan pattern that is constrained to a 2D plane as seen in Figure 5.9. Because the lidar being mounted on the front bumper of the robot as seen in Figure 5.3, the lidar scan will be obstructed by the robot itself, and was artificially limited to a FOV of 200°.

This lidar is outdoor capable, and can range up to 50m with reflectors however, an expectation without reflectors is roughly 20m at 25Hz with an angular resolution of 0.25°. Additionally, this lidar is quite accurate with a combined systematic and statistical error less than 5cm. Thus, the lidar is modeled as having error in both the range and angular measurements as seen in Equation (5.22).

Figure 5.8: SICK LMS151

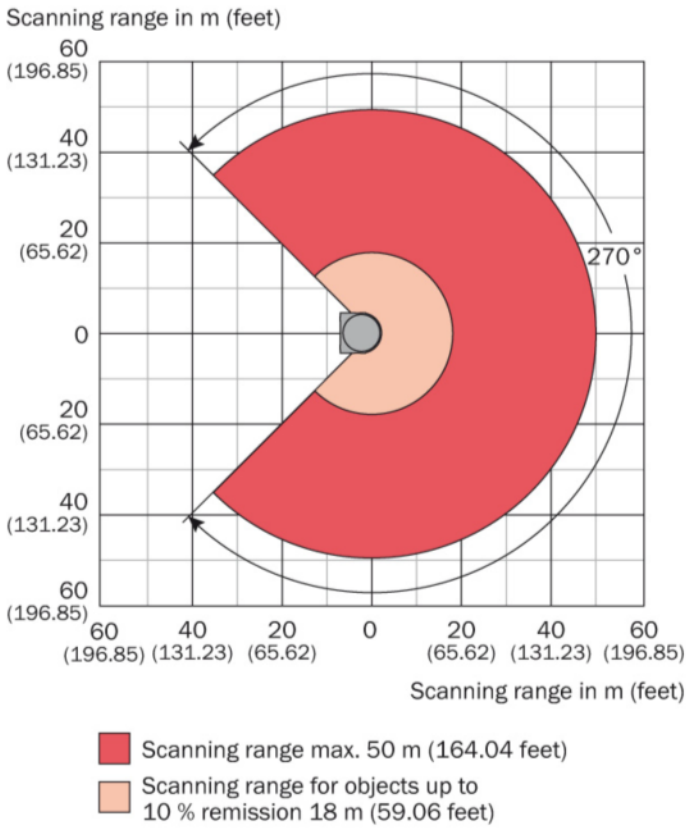

Figure 5.9: Scan Pattern

$$\begin{aligned} d_k &= \mathfrak{D}_k + w_d \quad w_d = N\left(0, \sigma_d^2\right) \\ \phi_k &= \vartheta_k + w_\phi \quad w_\phi = N\left(0, \sigma_\phi^2\right) \end{aligned} \tag{5.22}$$

Hence the true range ($\mathfrak{D}$) and angle (ϑ) are corrupted by gaussian noise. These range and angular measurements are defined in the lidar frame as seen in Figure 5.10, where the lidar is represented as a red trapezoid. Additionally because the lidar is scanning from effectively -100° to $+100^\circ$ the points are already ordered spatially by the lidar output itself.

5.3.3.2 Calculating Landmarks

The lidar is tasked with the critical job of finding landmarks. It was assumed that this robot would be primarily operating in highly structured environments such as offices or laboratories, and therefore due to the general lack of large complex geometric shapes in these environments, lines appear as an obvious choice for navigation. While there exist a number of different algorithms for extracting lines from 2D lidar scans in a structured environment, the work by [59] provides an excellent comparative study. The algorithm selected is known as "split and merge" (SAM), and provides an excellent balance of speed, correctness and ease of implementation. SAM is attributed as originating in [65], and has been used in a number of different applications [23, 79, 8, 96, 59]. As noted in Algorithm 5.1 [59], the method works by taking a complete scan of the environment, and letting this form a pool of points, which a best fit line is fit using total-least squares. The point with the largest distance from the line is selected. If the point is outside a distance threshold, the old line is split into two pools and the process repeats until this test is passed. Once the perpendicular distance test is passed, a number of sanity checks are performed on the line, such as a minimal number of points, a minimal line length requirement, and a maximum gap between points. If there are not enough points to form a line, or the line is not long enough, the list containing these points is removed from consideration. However, if the gap between points is too large, the pool of points is split at the gap, and the process repeats till all tests are passed. Finally,

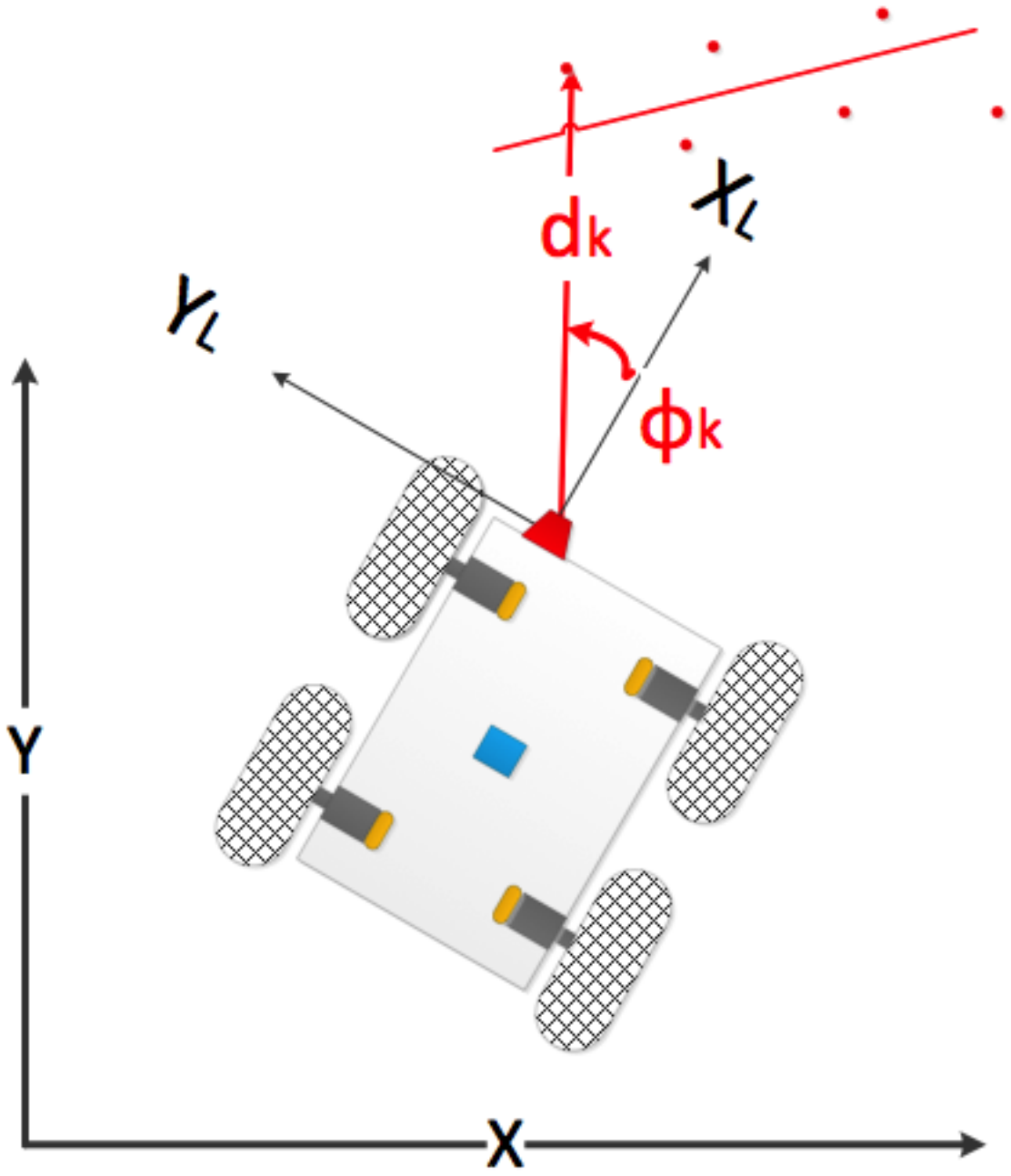

Figure 5.10: Lidar Measurement Definitions

Algorithm 5.1 Split and Merge (SAM)

```
1. Initial: set s1 consists of N points.  Put s1 in list L.
2. Fit a line using total least squares to the next s# in L.
3. Detect point P with a maximum distance d to the line.
4. If d is more than some threshold split s# into s#1 and s#2, replace s# in
5. If the length of the line is not at least J long, or of at least M points,
6. Detect consecutive points Pd1 and Pd2 which are of maximal distance betwee
If this distance is greater than threshold T, split s# into s#1 and s#2 at po
7. Determine covariance of line fit, and proceed to step 2 till L is exaughst
8. Merge collinear segments
```

a merging step occurs such that lines that represent a wall that are divided by a gap such as a doorway, can be treated as the same landmark. Note that with this algorithm it is important to work in the sequence that the points arrive, which best represent the structure of the environment, and not add points for consideration of a line randomly. Additionally, for this work, the largest gap between points allowed was 0.75 meters, with a maximum distance a point could be from a line of 0.0254 meters, and nine points where required to form a line.

Total least squares (TLS) was chosen as the regression technique of choice because it allows for errors in both axes of measurement, where ordinary least squares accounts for error in only one axis. The actual algorithm used is based heavily on the work of [67, 68]. Polar coordinates where selected to represent a line over cartesian coordinates, avoiding a situation where an estimated value would have to be infinite to try and represent the slope of a vertical line. Thus the regression is performed in a polar coordinate system, where the desired line parameters will be a range (ρ) and angle (α) that will be normal to line formed by the scan data, as seen in Figure 5.11. The scan data will also have components of range (d_k) and angle (ϕ_k) such that the algorithm will minimize that perpendicular distance δ_k from the scan points to the best fit line.

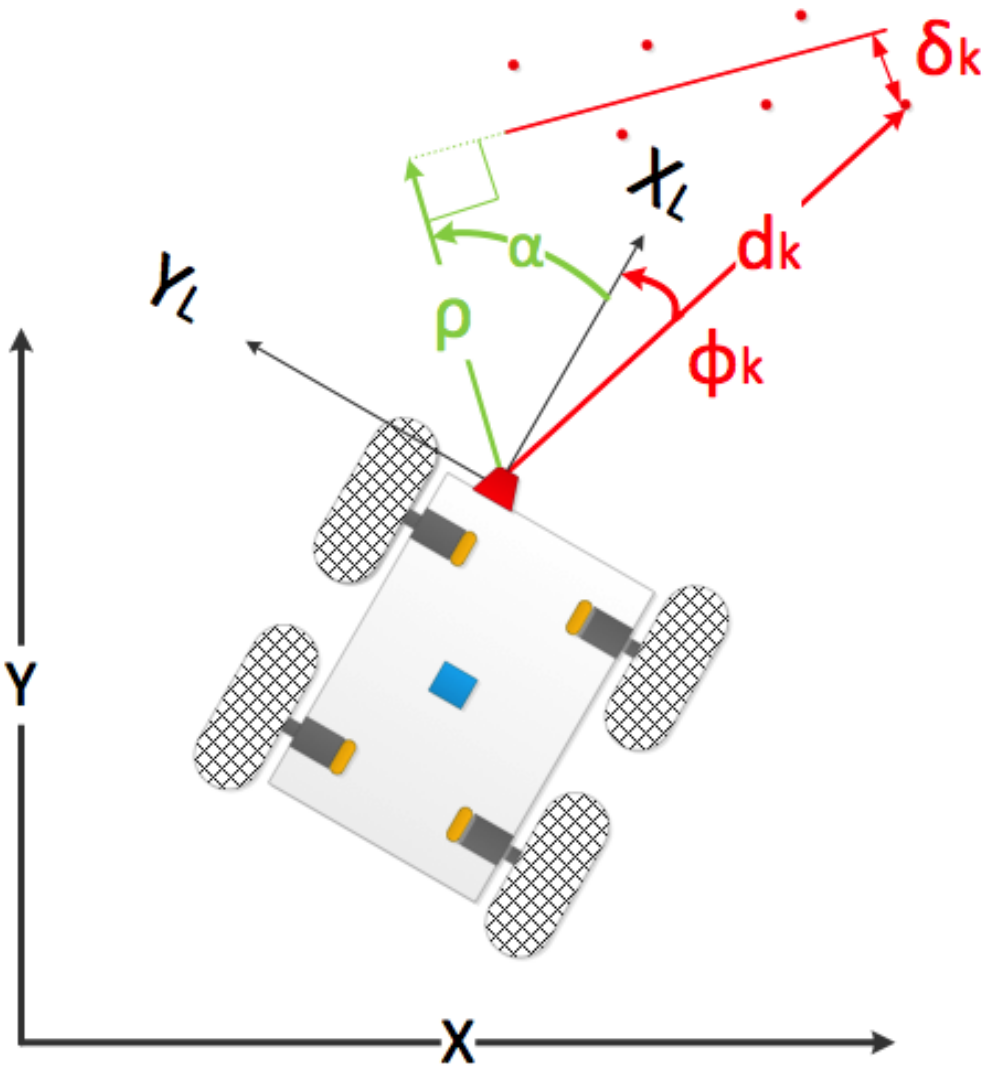

Figure 5.11: Landmark Regression

The algorithm used to perform TLS, is a recursive algorithm that requires an initial guess of the desired parameters $\hat{\rho}$ and $\hat{\alpha}$ which are obtained using traditional least squares. The value α is updated via Equation (5.23).

$$\alpha = \hat{\alpha} + \delta\alpha \tag{5.23}$$

Thus ρ can be obtained via Equation (5.24).

$$\rho = P_{\rho\rho}\left(\sum_{k=1}^{n} \frac{\cos\left(\hat{\alpha} - \phi_k\right)}{P_{\mathrm{TLS},k}}\right) \tag{5.24}$$

Where $P_{\rho\rho}$ and P_k are defined in Equations (5.25) and (5.26).

$$P_{\rho\rho} = \left(\sum_{k=1}^{n} P_{\mathrm{TLS},k}^{-1}\right)^{-1} \tag{5.25}$$

$$P_{\mathrm{TLS},k} = \sigma_d^2 \cos^2\left(\alpha - \vartheta_k^i\right) + \sigma_\phi^2 \left(\mathfrak{D}_k^i\right)^2 \sin^2\left(\alpha - \Phi_k^i\right) \tag{5.26}$$

Finally, $\delta\alpha$ can be calculated using Equation (5.27).

$$\delta\alpha = \frac{\sum_{k=1}^{n}\left(\frac{b_k a_k^{'} - a_k b_k^{'}}{(b_k)^2}\right)}{\sum_{k=1}^{n}\left(\frac{\left(a_k^{''} b_k - a_k b_k^{''}\right) b_k - 2\left(a_k^{\cdot} b_k - a_k b_k^{'}\right) b_k^{'}}{(b_k)^2}\right)} \tag{5.27}$$

such that

$$
\begin{aligned}
c_k &= \cos\left(\hat{\alpha} - \phi_k\right) \\
s_k &= \sin\left(\hat{\alpha} - \phi_k\right) \\
a_k &= \left(d_k c_k - \hat{\rho}^2\right) \\
a_k^{'} &= -2d_k^i s_k \left(d_k^i c_k - \hat{\rho}\right) \\
a_k^{''} &= 2\left(d_k^i\right)^2 s_k^2 - 2d_k^i c_k \left(d_k^i c_k - \hat{\rho}\right) \\
b_k &= \sigma^2 c_k^2 + \sigma_\phi^2 \left(d_k^i\right)^2 s_k^2 \\
b_k^{'} &= 2\left(\left(d_k^i\right)^2 \sigma_\phi^2 - \sigma_d^2\right) c_k s_k \\
b_k^{''} &= 2\left(\left(d_k^i\right)^2 \sigma_\phi^2 - \sigma_d^2\right)\left(c_k^2 - s_k^2\right)
\end{aligned}
\tag{5.28}
$$

Naturally, the process repeats for updating the value of α letting the knew ρ and α become $\hat{\rho}$ and $\hat{\alpha}$ until a stop condition is met, which was either a maximum of 15 iterations, or when $\delta\alpha$ was less than 0.0001°. Once the polar line parameters are calculated, the covariance of the fit is determined for eventual line merging, and incorporation into the SLACAM filter. The covariance will take the form seen in Equation (5.29).

$$
P_L = \begin{bmatrix} P_{\rho\rho} & P_{\rho\alpha} \\ P_{\alpha\rho} & P_{\alpha\alpha} \end{bmatrix} \tag{5.29}
$$

Where $P_{\rho\rho}$ has been previously defined in Equation (5.25), and $P_{\rho\alpha}$ and $P_{\alpha\alpha}$ are defined in Equations (5.30), and (5.31).

$$
P_{\rho\alpha} = P_{\alpha\rho} = \frac{P_{\rho\rho}}{G_T^{''}} \sum_{k=1}^{n} \left(\frac{2d_k^i \sin\left(\alpha - \phi_k^i\right)}{b_k} \right) \tag{5.30}
$$

$$
P_{\alpha\alpha} = \frac{1}{\left(G_T^{''}\right)^2} \sum_{k=1}^{n} \left(\frac{4\left(d_k^i\right)^2 \sin^2\left(\alpha - \phi_k^i\right)}{b_k} \right) \tag{5.31}
$$

Where

$$
G_T^{''} = \sum_{k=1}^{n} \left(\frac{\left(a_k^{''} b_k - a_k b_k^{''}\right) b_k - 2\left(a_k^{'} b_k - a_k b_k^{'}\right) b_k^{'}}{\left(b_k\right)^3} \right) \tag{5.32}
$$

using the definitions from Equation (5.28). Once this is completed for all extracted lines, collinear lines can be merged. This allows lines representing a wall that might be separated by a door opening to be treated as a single landmark. To accomplish this, each line is compared to all other lines in the set, such that if there are two lines as defined in Equation (5.33), it can be determined if they should be merged by performing the chi-squared test to see if the lines lie within a 3 sigma deviance threshold as defined by the covariance parameters previously determined.

$$L_1 = \begin{bmatrix} R_1 \\ \\ \alpha_1 \end{bmatrix} \qquad L_2 = \begin{bmatrix} R_2 \\ \\ \alpha_2 \end{bmatrix} \tag{5.33}$$

This chi-squared test is seen in Equation (5.34), and if passed, signifies that the lines are sufficiently similar statistically to be merged.

$$\chi^2 = \left(\begin{bmatrix} R_1 \\ \\ \alpha_1 \end{bmatrix} - \begin{bmatrix} R_2 \\ \\ \alpha_2 \end{bmatrix} \right)^T (P_{L_1} + P_{L_2})^{-1} \left(\begin{bmatrix} R_1 \\ \\ \alpha_1 \end{bmatrix} - \begin{bmatrix} R_2 \\ \\ \alpha_2 \end{bmatrix} \right) < 10^2 \tag{5.34}$$

If the lines pass the chi-squared test, they will be merged via Equations (5.35), and (5.36).

$$L_m = P_{L_m} \left((P_{L_1})^{-1} L_1 + (P_{L_2})^{-1} L_2 \right) \tag{5.35}$$

$$P_{L_m} = \left((P_{L_1})^{-1} + (P_{L_2})^{-1} \right)^{-1} \tag{5.36}$$

Note that the endpoints of the lines are not critical for navigation purposes, however they are retained for plotting and visualization purposes. If lines are merged at the scan level, their end points are still treated as representing two different lines, thus allowing better visualization of any gaps.

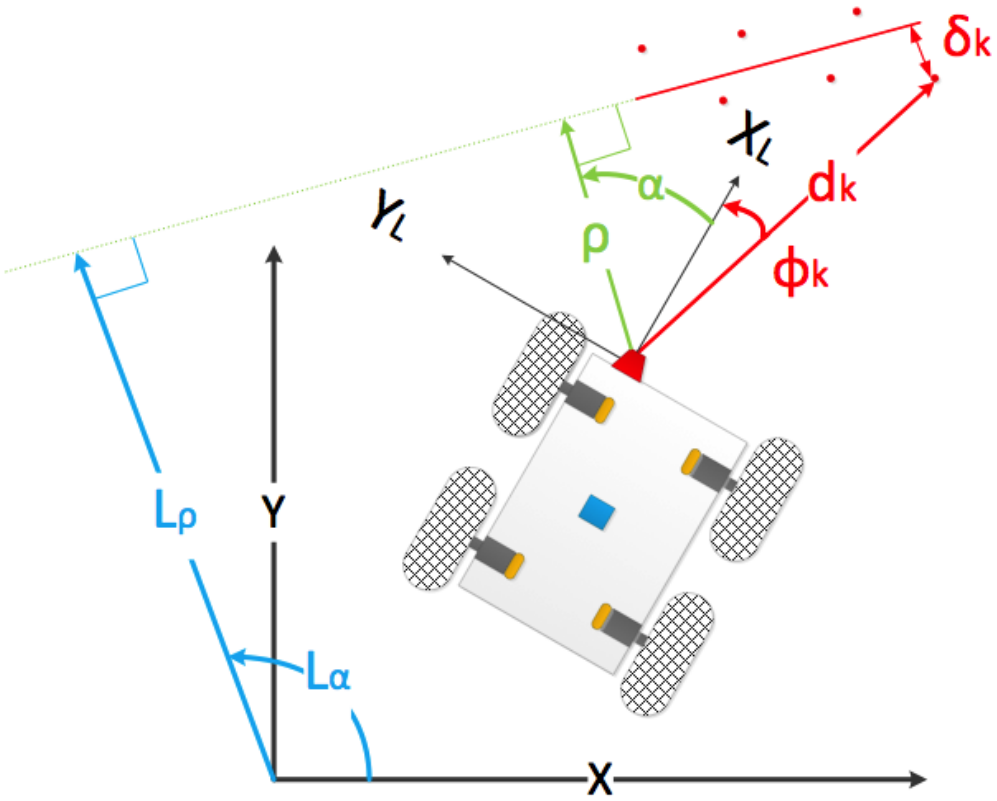

Figure 5.12: Line in Navigation Frame

5.3.3.3 Adding Landmarks to State Matrix

Before a landmark can be added into the state matrix, or used as a measurement, it must first be converted into the global navigation frame. Once again, polar coordinates will be used to represent the landmarks so that vertical lines can be represented with ease numerically. A landmark in the global frame will have some range (L_ρ) and angle (L_α), as defined by in Figure 5.12.

Note that despite the landmarks in both the lidar frame and the navigation frame being represented using polar coordinates, the conversion is performed by converting the parameters to cartesian coordinates, and back to polar. The endpoint of the normal vector in the lidar frame is defined in Equation (5.37).

$$\begin{aligned} x_{L,k}^{L} &= \rho_k \cos(\alpha_k) \\ y_{L,k} &= \rho_k \sin(\alpha_k) \end{aligned} \tag{5.37}$$

The parameters defined in Equation (5.37) are then converted into the IMU frame in Equation (5.38) using the rotation R_z as defined in Equation (5.9).

$$\begin{bmatrix} x_{L,k}^{b} \\ y_{L,k}^{b} \end{bmatrix} = R_z(\theta_L) \begin{bmatrix} x_{L,k}^{L} \\ y_{L,k}^{L} \end{bmatrix} + \begin{bmatrix} X_L \\ Y_L \end{bmatrix} \tag{5.38}$$

A similar procedure is then performed to obtain the parameters in the navigation frame as seen in Equation (5.39).

$$\begin{bmatrix} x_{L,k}^{n} \\ y_{L,k}^{n} \end{bmatrix} = R_z(\Theta) \begin{bmatrix} x_{L,k}^{b} \\ y_{L,k}^{b} \end{bmatrix} + \begin{bmatrix} X \\ Y \end{bmatrix} \tag{5.39}$$

The location of the lidar in the navigation frame is also calculated to provide a known point for using the point slope formula in future equations, which is calculated in Equation (5.40), where the subscript one is there only for future notational convenience.

$$\begin{bmatrix} X_{L,1}^{n} \\ Y_{L,1}^{n} \end{bmatrix} = R_z(\Theta) \begin{bmatrix} X_L \\ Y_L \end{bmatrix} + \begin{bmatrix} X \\ Y \end{bmatrix} \tag{5.40}$$

Based on simple geometry as seen in Figure 5.12, the line formed by L_ρ and L_α should have the same slope as the line formed by ρ and α. The slope is calculated in cartesian coordinates in Equation (5.41).

$$m = \frac{y_{L,k}^{L} - Y_{L,1}^{n}}{x_{L,k}^{L} - X_{L,1}^{n}} \tag{5.41}$$

The y-intercept of the line formed from the lidar to the point defined by ρ and α is calculated in Equation (5.42).

$$b = y_{L,k}^{L} + x_{L,k}^{L}\left(\frac{1}{m}\right) \tag{5.42}$$

The location for where L_ρ and L_α intersect the line defined by ρ and α in cartesian coordinates is calculated in Equation (5.43).

$$\begin{aligned} x_{\rho\alpha,L} &= b\left(m + \tfrac{1}{m}\right)^{-1} \\ y_{\rho\alpha,L} &= m\left(x_{\rho\alpha,L}\right) \end{aligned} \tag{5.43}$$

A simple conversion provides the desired line parameters in polar coordinates as seen in Equation (5.44).

$$\begin{aligned} L_\rho &= \sqrt{\left(x_{\rho\alpha,L}\right)^2 + \left(y_{\rho\alpha,L}\right)^2} \\ L_\alpha &= \text{atan2}\left(\tfrac{y_{\rho\alpha,L}}{x_{\rho\alpha,L}}\right) \end{aligned} \tag{5.44}$$

The function describing the conversion from the lidar frame to the navigation frame, will be defined by Equation (5.45), and is simply a function of the robot pose, the lidar calibration parameters, and the raw lidar measurements.

$$\begin{bmatrix} L_\rho \\ L_\alpha \end{bmatrix} = G_{L_\rho L_\alpha}\left(X,\ Y,\ \Theta,\ X_L,\ Y_L, \theta_L,\ \rho,\ \alpha\right) \tag{5.45}$$

The line line parameters can now be added to the state matrix as seen in Equation (5.46), where the newly added states are $L_{\rho,n+1}$ and $L_{\alpha,n+1}$.

$$\hat{X}_k = \begin{bmatrix} X \\ \vdots \\ \varsigma_R \\ L_{\rho_1} \\ L_{\alpha_1} \\ \vdots \\ L_{\rho_n} \\ L_{\alpha_n} \\ - - - \\ L_{\rho,n+1} \\ L_{\alpha,n+1} \end{bmatrix} = \begin{bmatrix} \hat{X}_k(1) \\ \vdots \\ \hat{X}_k(n) \\ - - - \\ G_{L_\rho L_\alpha}\left(X,\ Y,\ \Theta,\ X_L,\ Y_L, \theta_L,\ \rho,\ \alpha\right) \end{bmatrix} \tag{5.46}$$

Similarly, the covariance representing not just the line parameters, but how the line parameters relate to the other states, namely, the robot pose, the lidar calibration parameters, and the other landmarks will be added as defined in Equation (5.47).

$$\hat{P}_k = \begin{bmatrix} \hat{P}_k & | & C_P^{\mathrm{T}} \\ - & \cdot & - \\ C_P & | & C_{\rho\alpha} \end{bmatrix} =$$

$$\begin{bmatrix} \Sigma_{XX} & \Sigma_{YX} & \cdots & \Sigma_{\rho_n X} & \Sigma_{\alpha_n X} & | & \Sigma_{\rho_{n+1} X} & \Sigma_{\alpha_{n+1} X} \\ \Sigma_{XY} & \Sigma_{YY} & \cdots & \Sigma_{\rho_n Y} & \Sigma_{\alpha_n Y} & | & \Sigma_{\rho_{n+1} Y} & \Sigma_{\alpha_{n+1} Y} \\ \vdots & \vdots & \ddots & \vdots & \vdots & | & \vdots & \vdots \\ \Sigma_{X\rho_n} & \Sigma_{Y\rho_n} & \cdots & \Sigma_{\rho_n\rho_n} & \Sigma_{\alpha_n\rho_n} & | & \Sigma_{\rho_{n+1}\rho_n} & \Sigma_{\alpha_{n+1}\rho_n} \\ \Sigma_{X\alpha_n} & \Sigma_{Y\alpha_n} & \cdots & \Sigma_{\rho_n\alpha_n} & \Sigma_{\alpha_n\alpha_n} & | & \Sigma_{\rho_{n+1}\alpha_n} & \Sigma_{\alpha_{n+1}\alpha_n} \\ - & - & \cdots & - & - & \cdot & - & - \\ \Sigma_{X\rho_{n+1}} & \Sigma_{Y\rho_{n+1}} & \cdots & \Sigma_{\rho_n\rho_{n+1}} & \Sigma_{\alpha_n\rho_{n+1}} & | & \Sigma_{\rho_{n+1}\rho_{n+1}} & \Sigma_{\alpha_{n+1}\rho_{n+1}} \\ \Sigma_{X\alpha_{n+1}} & \Sigma_{Y\alpha_{n+1}} & \cdots & \Sigma_{\rho_n\alpha_{n+1}} & \Sigma_{\alpha_n\alpha_{n+1}} & | & \Sigma_{\rho_{n+1}\alpha_{n+1}} & \Sigma_{\alpha_{n+1}\alpha_{n+1}} \end{bmatrix} \tag{5.47}$$

The covariance of the best fit line in the lidar frame calculated in Equation (5.29), will be transformed into the global frame with the aid of Equation (5.45), as seen in Equation (5.48).

$$C_{\rho\alpha} = \left(\frac{\partial G_{L_\rho L_\alpha}}{\partial \hat{X}_k}\right) \hat{P}_k \left(\frac{\partial G_{L_\rho L_\alpha}}{\partial \hat{X}_k}\right)^{\mathrm{T}} + \left(\frac{\partial G_{L_\rho L_\alpha}}{\partial (\rho, \alpha)}\right) P_L \left(\frac{\partial G_{L_\rho L_\alpha}}{\partial (\rho, \alpha)}\right)^{\mathrm{T}} \tag{5.48}$$

Similarly, the cross terms are found using Equation (5.49).

$$C_P = \left(\frac{\partial G_{L_\rho L_\alpha}}{\partial \hat{X}_k}\right) \hat{P}_k \tag{5.49}$$

The raw lidar measurements in the form of point clouds can then be transformed into best fit lines whose line parameters and covariance can be transformed using the above equations into the navigation frame, and incorporated into the state matrix.

5.3.3.4 Using Landmarks as Measurements

In order for the line parameters extracted in Section 5.3.3.2 which where added to the state matrix in Section 5.3.3.3, to be of use for navigation, there must exist a way to use them as navigational measurements. This inherently relies on the premise that the previous landmarks that where added to the state matrix, will be observed again at a future time, a basic tenant of SLACAM. When a line is to be used for navigation, the newly scanned line must be associated with an existing line, and from there, a prediction or expectation of the line parameters based on this association must be created for use in the EKF. To determine which line to associate the newly scanned line with, it will be assumed that the line has already been converted into the navigation frame using Equation (5.45). Once this conversion is complete the association of the converted line to a line in the current state matrix is found using the mahalanobis distance (D_M), such that the newly converted line is compared against all lines currently existing in the state matrix as seen in Equation (5.50).

$$
\begin{aligned}
dz &= \left(\begin{bmatrix} \rho_n \\ \alpha_n \end{bmatrix} - \begin{bmatrix} L^L_{\rho,n} \\ L^L_{\alpha,n} \end{bmatrix} \right) \\
S &= (H_L)\, \hat{P}_k\, (H_L)^{\mathrm{T}} + P_L \\
D_M &= \sqrt{dz^{\mathrm{T}} S^{-1} dz}
\end{aligned}
\tag{5.50}
$$

Note this equation relies on converting the lines in the current state matrix into the lidar frame which are denoted as L^L_ρ and L^L_α, where H_L is the linearization of this conversion. The line in the state matrix which has the minimum mahalanobis distance will be the line that is best associated with the newly converted line. If the mahalanobis distance is below a threshold of $\text{merge}_{\min}$ then the lines are deemed closely related enough to be used for navigation. If the distance is above a threshold of $\text{merge}_{\max}$, then the newly scanned line is deemed unique enough to a be new landmark, and instead of being used to navigate during this iteration of the filter, it will be added to the state matrix through the process shown in Section 5.3.3.3. If the mahalanobis distance is between these two thresholds, then it is considered too ambiguous and the newly scanned line is simply not used for navigational purposes. This process repeats for every line extracted from the lidar for the current scan.

Before the Mahalanobis distance can be found, the current landmarks in the state matrix must first be converted into the lidar frame, which will eventually be linearized to obtain H_L. To convert the line from the navigation frame to the lidar frame, the location of the lidar in the navigation frame must first be found via Equation (5.51).

$$
\begin{bmatrix} x^n_L \\ y^n_L \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} + R_z\,(\Theta) \begin{bmatrix} X_L \\ Y_L \end{bmatrix} \tag{5.51}
$$

The angle to the lidar is defined as a_p as defined in Figure 5.13, and is calculated in Equation (5.52).

$$
a_p = \text{atan2}\left(\frac{y^n_L}{x^n_L} \right) \tag{5.52}
$$

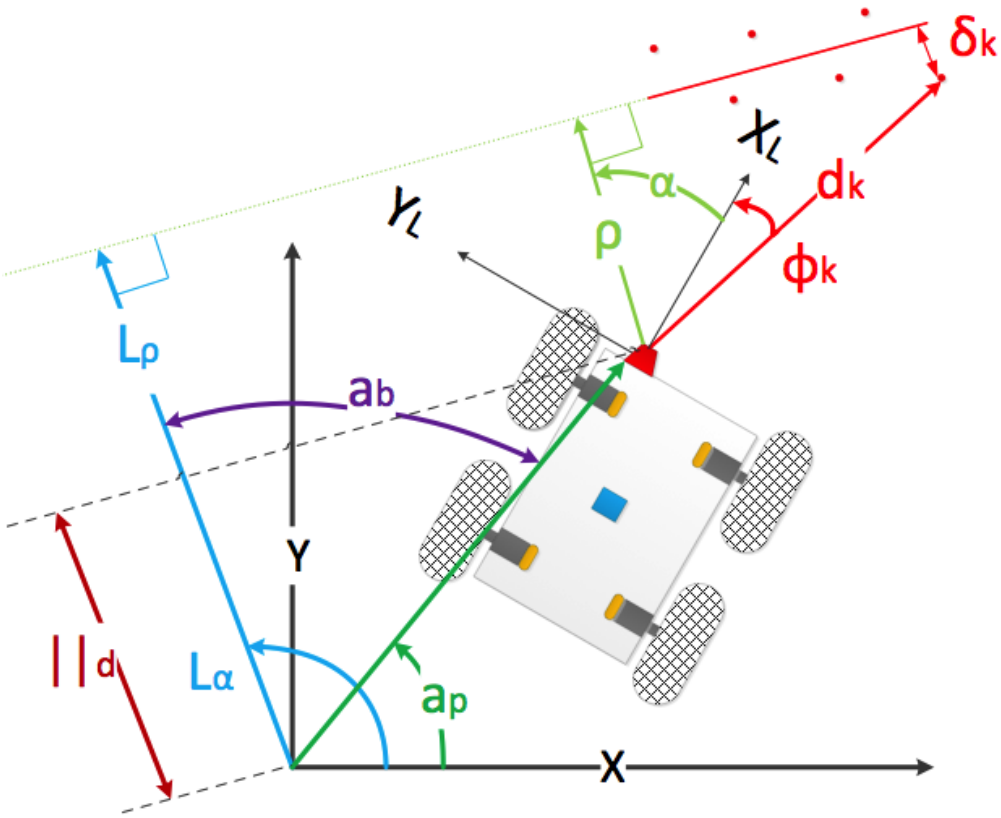

Figure 5.13: Global to Local

The angle between global landmark and the angle to the lidar is defined as a_b as calculated in Equation (5.53).

$$a_b = a_p - L_\alpha \tag{5.53}$$

The distance defined as $||_d$ which is the distance that is parallel distance between the global landmark and the distance to the lidar is defined in Equation (5.54).

$$||_d = \sqrt{\left(x_L^n\right)^2 + \left(y_L^n\right)^2} \cos\left(a_b\right) \tag{5.54}$$

Thus, the predicted distance perceived by the lidar is defined in Equation (5.55), which in Figure 5.13, would be the same distance as defined by ρ.

$$L_\rho^L = L_\rho - ||_d \tag{5.55}$$

The predicted angle to the landmark in the lidar frame is defined in Equation (5.56), which in Figure 5.13, would be the same angle as that defined as α.

$$L_\alpha^L = R_z\left(L_\alpha - \Theta - \theta_L\right) ||_d \tag{5.56}$$

The conversion process is a function of the robot location, the lidar mounting parameters, and the landmark to be converted, which is defined in Equation (5.57).

$$\begin{bmatrix} L_\rho^L \\ L_\alpha^L \end{bmatrix} = G_L\left(X,\ Y,\ \Theta,\ X_L,\ Y_L,\ \theta_L,\ L_\rho^n,\ L_\alpha^n\right) \tag{5.57}$$

Finally, G_L is linearized via Equation (5.58), which equates to the term H_L as needed for Equation (5.50).

$$H_L = \frac{\partial}{\partial \hat{X}_k} G_L \tag{5.58}$$

The equations necessary to convert the current landmarks into the lidar frame as well as determine which lines extracted from a lidar scan correspond to which landmark have been established. Note that the matrix Z_{LP} is defined in Equation (5.59),

$$Z_{LP} = \left[\left[\begin{array}{c} \rho_1 \\ \alpha_1 \end{array} \right]^{\mathrm{T}}, \cdots, \left[\begin{array}{c} \rho_n \\ \alpha_n \end{array} \right]^{\mathrm{T}} \right]^{\mathrm{T}} \tag{5.59}$$

which is simply an array of lines extracted from the lidar that had a mahalanobis distance of at least merge_{min}, and Equation (5.60) defines an array of landmarks to which Equation Z_{LP} corresponds.

$$G_{LP} = \left[\left[\begin{array}{c} L^L_{\rho_1} \\ {L^L}_{\alpha_1} \end{array} \right]^{\mathrm{T}}, \cdots, \left[\begin{array}{c} L^L_{\rho_n} \\ {L^L}_{\alpha_n} \end{array} \right]^{\mathrm{T}} \right]^{\mathrm{T}} \tag{5.60}$$

Therefore, Z_{LP} essentially defines measurements to existing landmarks, and G_{LP} defines a prediction of the lidar measurements to that landmark.

5.3.3.5 Merging Landmarks

Merging of landmarks in the global frame occurs as the final step in the loop, after the measurement update. The need to merge landmarks in the global frame arises because landmarks were not sufficiently similar to one another in the landmark frame to be merged at the time, but as successive observations have occurred, it has become apparent that they are indeed attempting to represent the same landmark. Merging landmarks has the benefit of both increasing the fidelity of the surrounding environment and the confidence of the map, as well as the additional benefit of reducing the state matrix. While it is possible to perform merging on both a statistical as well as position based or line overlap basis, it was found that performing line merging based purely on statistical values yielded the best results. Note that this test is exhaustive, such that each landmark is the state matrix, is compared against

every other existing landmark in the state matrix. Hence the chi-squared test is applied on the two potential landmarks to be merged as calculated in Equation (5.61).

$$\begin{aligned} \delta L &= \begin{bmatrix} L_{\rho,i} \\ L_{\alpha,i} \end{bmatrix} - \begin{bmatrix} L_{\rho,j} \\ L_{\alpha,j} \end{bmatrix} \\ P_{L,(i,j)} &= \begin{bmatrix} \Sigma_{\rho_{(i,j)}\rho_{(i,j)}} & \Sigma_{\alpha_{(i,j)}\rho_{(i,j)}} \\ \Sigma_{\rho_{(i,j)}\alpha_{(i,j)}} & \Sigma_{\alpha_{(i,j)}\alpha_{(i,j)}} \end{bmatrix} \\ \chi^2 &= (\delta L)^{\mathrm{T}} \left(P_{L,i} + P_{L,j}\right)^{-1} (\delta L) \end{aligned} \tag{5.61}$$

If the chi-squared value is below a given threshold, then the the two landmarks will be merged according to Equations (5.62) and (5.63).

$$P_{L,\text{new}} = \left(P_{L,i}^{-1} + P_{L,j}^{-1}\right)^{-1} \tag{5.62}$$

$$\begin{bmatrix} L_{\rho,\text{new}} \\ L_{\alpha,\text{new}} \end{bmatrix} = P_{L,\text{new}} \left(P_{L,i}^{-1} \begin{bmatrix} L_{\rho,i} \\ L_{\alpha,i} \end{bmatrix} + P_{L,j}^{-1} \begin{bmatrix} L_{\rho,j} \\ L_{\alpha,j} \end{bmatrix} \right) \tag{5.63}$$

The new covariance and landmark values are substituted for one of the old landmark values, and the values corresponding to the other landmark are simply removed from the state matrix.

5.4 Lidar to IMU Calibration

The lidar to IMU calibration is performed by comparing the lidar change in pose with that of the IMU change in pose. Additionally, the velocities are compared as an additional aiding measurement. The change in lidar pose is determined using ICP as outlined in Section 3.3, where successive lidar scans are compared to determine the change in pose of the lidar from scan to scan, which are defined as δX_{lidar}, δY_{lidar}, $\delta\Theta_{\text{lidar}}$. However, because of the relatively slow rate of motion compared with the scanning rate of the lidar, the individual

points of the lidar scan are uncompensated for vehicle motion. To obtain a measurement of the velocity and rotation rate of the lidar, the change in pose is simply divided by the sample rate of the lidar, as defined in Equation (5.64).

$$\begin{bmatrix} v_x^l \\ v_y^l \\ \omega_z^l \end{bmatrix} = \begin{bmatrix} \delta X_{\text{lidar}}/dt_{\text{lidar}} \\ \delta Y_{\text{lidar}}/dt_{\text{lidar}} \\ \delta \Theta_{\text{lidar}}/dt_{\text{lidar}} \end{bmatrix} \tag{5.64}$$

Note that all measurements are understood to be in the lidar frame.

Obtaining relative pose of the IMU consists of first treating the location of the IMU during the previous lidar scan as the origin. The IMU is then integrated over the lidar sample time, using a history of the previous IMU measurements over this time period. The actual integration is accomplished using methods similar to those outlined in Section 5.3.1, however only the position and velocity states are analyzed. Hence, the IMU input is defined in Equation (5.65),

$$u(\tau) = \begin{bmatrix} \widetilde{A}_x(\tau) & \widetilde{A}_y(\tau) & \widetilde{G}_z(\tau) \end{bmatrix}^{\text{T}}, \; u(\tau) \subset [t - dt_{\text{ lidar}}, t] \tag{5.65}$$

where t represents only the range from the previous to the current lidar measurement. The actual states to be integrated and the corresponding update equation is defined in (5.66).

$$\dot{\hat{\mathbf{X}}}(\tau) = \begin{bmatrix} \dot{X}(\tau) \\ \dot{Y}(\tau) \\ \dot{\Theta}(\tau) \\ - - - \\ \ddot{X}(\tau) \\ \ddot{Y}(\tau) \end{bmatrix}, \; F\left(\hat{\mathbf{X}}, \; \mu\right) = \begin{bmatrix} \dot{X}(\tau - 1) \\ \dot{Y}(\tau - 1) \\ \widetilde{G}_z(\tau) - b_z \\ - - - \\ C_b^n \begin{bmatrix} \widetilde{A}_x(\tau) - b_x \\ \widetilde{A}_y(\tau) - b_y \end{bmatrix} \end{bmatrix} \tag{5.66}$$

The states are integrated using fourth order Runge-Kutta as seen in Equation (5.67),

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_{k-1} + \tfrac{1}{6}K_1 + \tfrac{1}{3}K_2 + \tfrac{1}{3}K_3 + \tfrac{1}{6}K_4 \tag{5.67}$$

where K is defined in equation (5.68). Note that the bias states are allowed to remain constant during this brief integration, and the latest bias estimates from the filter are used.

$$\begin{aligned} K_1 &= F\left(\hat{\mathbf{X}}_{k-1},\ \mu\right) dt_{\text{imu}} \\ K_2 &= F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_1}{2},\ \mu\right) dt_{\text{imu}} \\ K_3 &= F\left(\hat{\mathbf{X}}_{k-1} + \tfrac{K_2}{2},\ \mu\right) dt_{\text{imu}} \\ K_4 &= F\left(\hat{\mathbf{X}}_{k-1} + K_3,\ \mu\right) dt_{\text{imu}} \end{aligned} \tag{5.68}$$

By allowing the initial position position to be zero, and the velocities of the robot to be rotated into the body frame as seen in Equation (5.69), the change in pose of the IMU of the lidar measurement period will simply be the pose states of $\hat{X}(\tau)$.

$$\hat{X}\left(t^- - dt_{\text{lidar}}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ C_n^b \begin{bmatrix} \dot{X} & \dot{Y} \end{bmatrix}^{\mathrm{T}} \end{bmatrix} \tag{5.69}$$

Finally, the rotation rate of the IMU is simply defined as the current rotation rate as measured by the IMU minus the current bias estimate. Thus, the entire estimate for the change in pose and velocity of the IMU which will be defined as $\hat{I}$ is provided in Equation (5.70) and is a function of both the current state matrix and the raw IMU measurements.

$$\hat{I}\left(\hat{\mathbf{X}},\ \mu\right) = \begin{bmatrix} \delta X_{\text{imu}} \\ \delta Y_{\text{imu}} \\ \delta \Theta_{\text{imu}} \\ v_x^b \\ v_y^b \\ \omega_z^b \end{bmatrix} = \begin{bmatrix} X\left(\tau\right) \\ Y\left(\tau\right) \\ \Theta\left(\tau\right) \\ \dot{X}\left(\tau\right) \\ \dot{Y}\left(\tau\right) \\ \widetilde{G}_z(\tau) - b_z \end{bmatrix} \tag{5.70}$$

In order for the velocity and the change in pose of the lidar to be compared to the IMU, they first must be resolved into the same frame. The change in the IMU measurements from $\hat{I}$ will be transformed using equation (3.5), which is seen below in Equation (5.71) with the necessary notational alterations which includes the velocity terms for the 2D case.

$$G_{LiC}\left(\hat{X}_k,\ \mu\right) = \begin{bmatrix} \hat{\delta}^{t,L}_{t-dt_{\text{lidar}},i} = R_i^L\left(\delta^{t,i}_{t-dt_{\text{lidar}},i} + R^{t,i}_{t-dt_{\text{lidar}},i} S_\circ^L - S_\circ^L\right) \\ \hat{R}^{t,L}_{t-dt_{\text{lidar}},L} = R^{t,i}_{t-dt_{\text{lidar}},i} \\ \hat{\dot{\delta}}^{t,L}_{t-dt_{\text{lidar}},i} = R_i^L \dot{\delta}^{t,i}_{t-dt_{\text{lidar}},i} + \dot{R}^{t,i}_{t-dt_{\text{lidar}},i} \times R_i^L S_\circ^L \\ \hat{\dot{R}}^{t,L}_{t-dt_{\text{lidar}},L} = \dot{R}^{t,i}_{t-dt_{\text{lidar}},i} \end{bmatrix} \tag{5.71}$$

Note the terms used in Equation (5.71) are defined as follows.

$$R_i^L = \left(R_L^i\right)^{\text{T}} = \begin{bmatrix} \cos\left(\theta_L\right) & \sin\left(\theta_L\right) \\ -\sin\left(\theta_L\right) & \cos\left(\theta_L\right) \end{bmatrix} \tag{5.72}$$

$$S_\circ^L = \begin{bmatrix} X_L & Y_L \end{bmatrix}^{\text{T}} \tag{5.73}$$

$$R^{t,i}_{t-dt_{\text{lidar}},i} = \delta \Theta_{\text{imu}} \tag{5.74}$$

$$\dot{R}^{t,i}_{t-dt_{\text{lidar}},i} = \omega_z^b \tag{5.75}$$

Additionally G_{LiC} from Equation (5.71) will constitute the prediction of the lidar change in pose and velocity of the measurement z_{LiC} defined in Equation (5.76)

$$Z_{LiC} = \begin{bmatrix} \delta X_{\text{lidar}} & \delta Y_{\text{lidar}} & \delta \Theta_{\text{lidar}} & v_x^l & v_y^l & \omega_z^l \end{bmatrix}^{\text{T}} \tag{5.76}$$

Finally the linearization of the prediction G_{LiC} will be defined as H_{LiC}, defined in Equation (5.77).

$$H_{LiC} = \frac{\partial}{\partial \hat{X}_k} G_{LiC}\left(\hat{X}_k,\ \mu\right) \tag{5.77}$$

It is worth noting that G_{LiC} specifically relies on the yaw, inertial biases, and velocities of the robot as well as the mounting parameters of the lidar. Additionally, even if the yaw and velocity information where not available, sufficient information would still exist for calibration knowing only the relative change in pose. Hence, the equations necessary to calibrate the IMU to the lidar in 2D using relative pose and velocity measurements has been developed.

5.5 IMU to Odometry Calibration via Lidar

The IMU to odometry calibration determines the necessary odometry needed for accurate navigation. Specifically those parameters listed in Equation (5.78), as first seen in Section 3.5, which constitute the instantaneous centers of rotation and the wheel radii.

$$\begin{bmatrix} x_{ICR} & y_{ICR_l} & y_{ICR_r} & R_l & R_r \end{bmatrix} \tag{5.78}$$

Recall from Section 5.3.2, that the odometry values are being used directly to help compensate the IMU inaccuracies and biases. Immediately attempting to use the IMU to assist the odometry measurement would appear pedantic, especially since these sensors are both noisy and not highly accurate. In order to determine the IMU to odometry calibration values, the lidar will be used to compare relative pose measurements, and the result will be transformed

into the IMU frame. The same change in pose measurements obtained from ICP used in the Section 5.4 for the lidar to IMU calibration, will be also be used here and are reproduced from Equation (5.76) below.

$$Z_{LiC} = \begin{bmatrix} \delta X_{\text{lidar}} & \delta Y_{\text{lidar}} & \delta\Theta_{\text{lidar}} & v_x^l & v_y^l & \omega_z^l \end{bmatrix}^{\text{T}} \tag{5.79}$$

The change in pose and velocities of the vehicle at the IMU are then calculated as seen in Equation (5.80) using the previously developed Equation (3.27).

$$\begin{bmatrix} \delta_x^b \\ \delta_y^b \\ \delta\Theta_z^b \\ - - - \\ v_x^b \\ v_y^b \\ \omega_z^b \end{bmatrix} = \begin{bmatrix} \frac{1}{\left(y_{ICR_l} - y_{ICR_r}\right)} \begin{bmatrix} -R_l y_{ICR_r} & R_r y_{ICR_l} \\ -R_l x_{ICR} & R_r x_{ICR} \\ -R_l & R_r \end{bmatrix} \begin{bmatrix} \sum Odom_l(t) - Odom_l(t - dt_{\text{lidar}}) \frac{2\pi}{tpr} \\ \sum Odom_r(t) - Odom_r(t - dt_{\text{lidar}}) \frac{2\pi}{tpr} \end{bmatrix} \\ - - - \\ \frac{1}{\left(y_{ICR_l} - y_{ICR_r}\right)} \begin{bmatrix} -R_l y_{ICR_r} & R_r y_{ICR_l} \\ -R_l x_{ICR} & R_r x_{ICR} \\ -R_l & R_r \end{bmatrix} \begin{bmatrix} \frac{\sum Odom_l(t) - Odom_l(t - dt_{\text{lidar}})}{tpr} \frac{2\pi}{dt_{\text{lidar}}} \\ \frac{\sum Odom_r(t) - Odom_r(t - dt_{\text{lidar}})}{tpr} \frac{2\pi}{dt_{\text{lidar}}} \end{bmatrix} \end{bmatrix} \tag{5.80}$$

These values are then transformed into the lidar frame using the process outlined in Equation (5.71) which is reproduced below in Equation (5.81) with notational alterations necessary for the odometry calibration.

$$G_{LoC}\left(\hat{X}_k,\ \mu\right) = \begin{bmatrix} \hat{\delta}_{t-dt_{\text{lidar}},b}^{t,L} = R_b^L \left(\delta_{t-dt_{\text{lidar}},b}^{t,b} + R_{t-dt_{\text{lidar}},b}^{t,b} S_\circ^L - S_\circ^L\right) \\ \hat{R}_{t-dt_{\text{lidar}},L}^{t,L} = \delta\Theta_z^b \\ \hat{\dot{\delta}}_{t-dt_{\text{lidar}},b}^{t,L} = R_i^L \dot{\delta}_{t-dt_{\text{lidar}},b}^{t,b} + \omega_z^b \times R_b^L S_\circ^L \\ \hat{\dot{R}}_{t-dt_{\text{lidar}},L}^{t,L} = \omega_z^b \end{bmatrix} \tag{5.81}$$

The necessary support equations are listed in Equation (5.82) and (5.83).

$$R_b^L = \left(R_L^b\right)^{\mathrm{T}} = \begin{bmatrix} \cos(\theta_L) & \sin(\theta_L) \\ -\sin(\theta_L) & \cos(\theta_L) \end{bmatrix} \tag{5.82}$$

$$S_\circ^L = \begin{bmatrix} X_L & Y_L \end{bmatrix}^{\mathrm{T}} \tag{5.83}$$

Equation (5.71) provides the necessary prediction of the change in lidar pose and velocity by transforming the odometry measurements into the IMU frame, and then into the lidar frame. The linearization of this transformation is defined in Equation (5.84).

$$H_{LoC} = \frac{\partial}{\partial \hat{X}_k} G_{LoC}\left(\hat{X}_k,\ \mu\right) \tag{5.84}$$

Note that G_{LoC} is a function of lidar mounting parameters, robot motion, and the odometry calibration parameters.

5.6 Measurement Update Recap

Because of the host of different measurement types, sensors, and scenarios when different updates are performed, this section will both recap the previously mentioned measurement updates and provide greater insight into the measurement update process. As seen in Section 3.2, the Kalman filter measurement update is repeated below in Equation (5.85).

$$\begin{aligned} y_k &= z_k - G_k \\ K_k &= P_k^- H_k^{\mathrm{T}} \left(H_k P_k^- H_k^{\mathrm{T}} + R_k\right)^{-1} \\ \hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_{\mathbf{k}}^- + K_k y_k \\ P_k &= \left(I - K_k H_k\right) P_k^- \end{aligned} \tag{5.85}$$

While there are three sensors, there are two primary types of measurement updates, the first is a wheel odometry based update, and the other is a lidar based update. The odometry based measurement update occurs at the rate of the wheel odometry sensor and comprises

the mutually exclusive measurements of either a motion update as defined in Equations (5.17-5.19), or a zero velocity update as defined in Equations (5.20-5.21), as demonstrated in Equation (5.86).

$$y_{k,\text{odom}} = \begin{bmatrix} Z_{k,\text{odom}} \\ --- \\ Z_{k,\text{zupt}} \end{bmatrix} - \begin{bmatrix} G_{k,\text{odom}} \\ --- \\ G_{k,\text{zupt}} \end{bmatrix} \tag{5.86}$$

The lidar based update provides the bulk of the navigation and calibration data. The lidar based update occurs at the rate of the lidar and can incorporate any, all, or none of the various sub matrices which are comprised of the wheel odometry calibration using Equations (5.76) and (5.81), the lidar to IMU calibration using Equations (5.71) and (5.76), and the necessary landmark updates using Equations (5.59) and (5.60) as denoted in Equation (5.87).

$$y_{k,\text{lidar}} = \begin{bmatrix} Z_{LiC} \\ Z_{LiC} \\ Z_{LP} \end{bmatrix} - \begin{bmatrix} G_{LoC} \\ G_{LiC} \\ G_{LP} \end{bmatrix} \tag{5.87}$$

Note, only after a potential lidar update are new landmarks added to the state matrix.

The lidar calibration will not occur unless the robot has determined to have moved a sufficient amount. Similarly odometry calibration will not occur unless the wheel odometry has determined that the robot is moving and the lidar has moved sufficiently. Finally, the landmark based update will not occur if no landmarks are currently in the state-matrix or no landmarks where detected for a particular scan, however this will does not inherently affect calibration.

5.7 Observability Analysis

Before this SLACAM technique is validated through simulation and experimentation, the observability of this problem formulation is assessed below. Specifically, the observability

will be analyzed using the Fischer information matrix (FIM). This analysis will build upon previous analysis using the FIM which proved range-bearing SLAM was observable [90], and that of calibration through the use of incremental pose [11, 12] where SLAM was treated as a relative pose problem instead of one based on global accuracy. This analysis, will adopt the same assumptions as the previous stated works as well as that more than one non-collinear landmark will be present. The equation for the iterative FIM as initially seen in Equation (3.34) is reproduced below.

$$J_k = \left(\Phi_{k-1}^{-1}\right)^{\mathrm{T}} J_{k-1} \Phi_{k-1}^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \tag{5.88}$$

Note that for values not affected by the time update such as the location of landmarks and the lidar and odometry calibration parameters, the FIM will provide an assessment of the observability of those parameters regardless of the robot trajectory provided that J_k obtains full column rank. However, the observability of parameters such as position and velocity, will be predicated on the robot trajectory. It is possible for SLACAM to operate successfully even when J_k never obtains full rank. For example, this would occur when a landmark was added to the state matrix which was never observed, merely implying that this landmark was not observable, and not the entire SLACAM formulation. Additionally, the necessary rank of J_k can drop over time if landmarks are merged together. The rank of J_k is seen graphically in Figure 5.14 though the first several iterations of the SLACAM algorithm, from which it can be seen that every state becomes observable over the entire length of the trajectory. Note that full observability is obtained during iteration 1026. Before this, there has been a rank deficiency because the robot has had insufficient motion to calibrate the lidar and odometry states. Specifically, the robot has been siting motionless. However, once the robot motion is detected, these states are estimated, and full rank is obtained. Notice that the rank of J_k increases around iteration 1200 before dropping back down at iteration 1300. A landmark is being added which causes the desired rank to increase and the desired

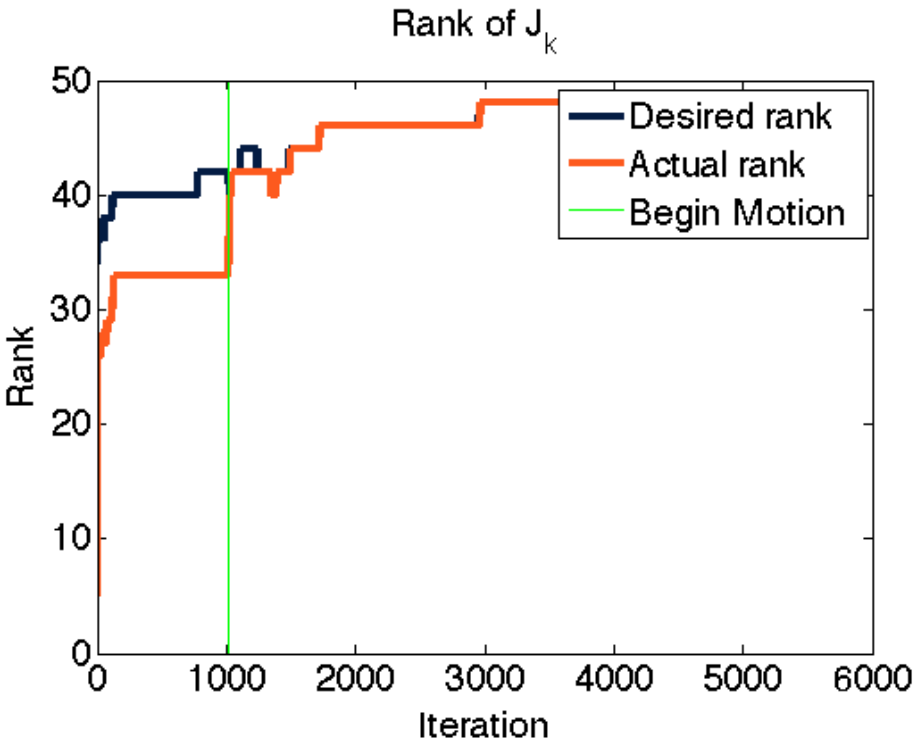

Figure 5.14: Rank of J_k

rank decreases when that landmark is merged with an already existing landmark. A similar change in rank is observed near iteration 1400 when the desired rank drops and quickly returns. This is also due to landmarks being merged, and then added. Similarly, the desired rank slowly increases as landmarks are added.

Therefore the lidar and odometry calibration parameters as well as the landmark locations are shown to be observable. Additionally, it is shown that enough information exists for the position, velocity, and IMU bias parameters to become observable. However, because these parameters are trajectory based, the rank of J_k must be reset when it achieves full rank to assess the observability of these parameters. This reseting of J_k is seen in Figure 5.15.

Note that J_k is rank deficient immediately after being reset, however full rank is achieved quickly during motion. This motion requirement conforms to the previous auto-calibration analysis performed for the 3D case in Chapter 3, which indicated that some change in rotation

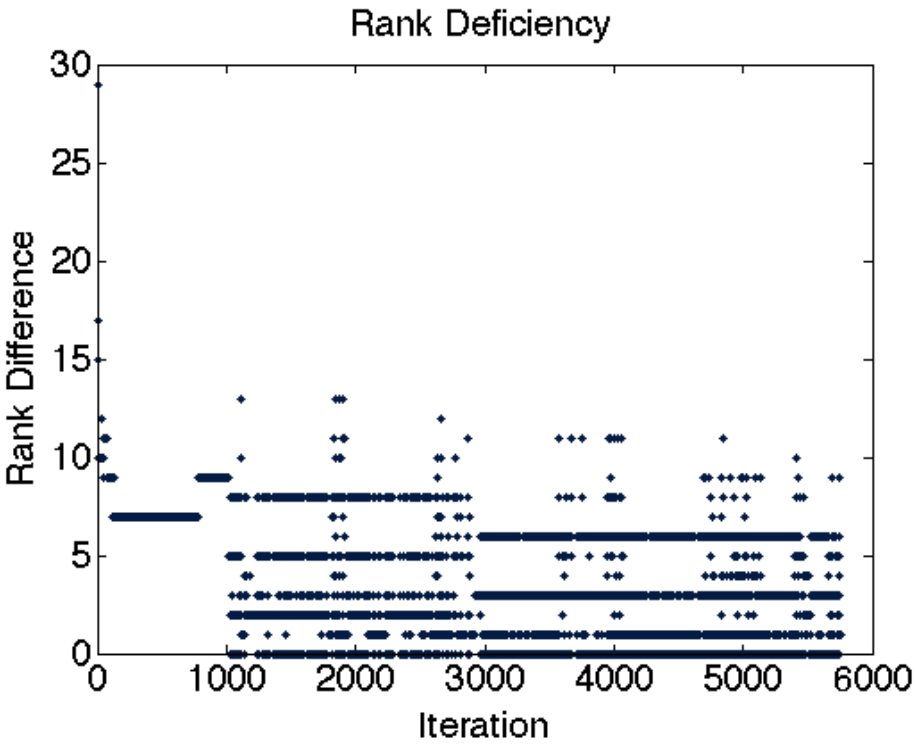

Figure 5.15: Trajectory Analysis

was necessary for calibration. Thus it has been shown that sufficient information exists for SLACAM to be observable. Additionally, by virtue of full rank being achieved this implies that sufficient excitation existed, hence a non degenerate path was taken.

Chapter 6

Validation of SLACAM Technique

This chapter will validate the SLACAM technique through both simulated and experimental results. The overview of the simulation setup and results will be presented in Sections 6.1 and 6.2, which will focus on the ability to correctly calibrate the sensor parameters, construct an accurate map, as well as return to the starting location correctly. Similar analysis will be performed on the experimental data as will be overviewed in Sections 6.3 and 6.4, where the focus will be on calibration accuracy and improvement, map creation, as well as accuracy in returning to the starting location.

6.1 Simulated Setup

This section outlines the creation of the simulated robot, environment, path, and sensor outputs. Naturally the simulated robot, sensors, and environment were constructed with the future experimental validation in mind. The following subsections detail the specific of the simulated environment.

6.1.1 Generating Map

Because the landmark detection algorithm is line based, it will inherently perform better in structured environments. The second floor of the Advanced Research Building was selected as the simulation environment due to ease of access as well as it being constructed as a structured environment composed of relatively straight walls. Additionally, basing the map off this environment lends additional credence to the validity of the simulation environment as well as having long hallways with few features - a know SLACAM/SLAM weakness. The

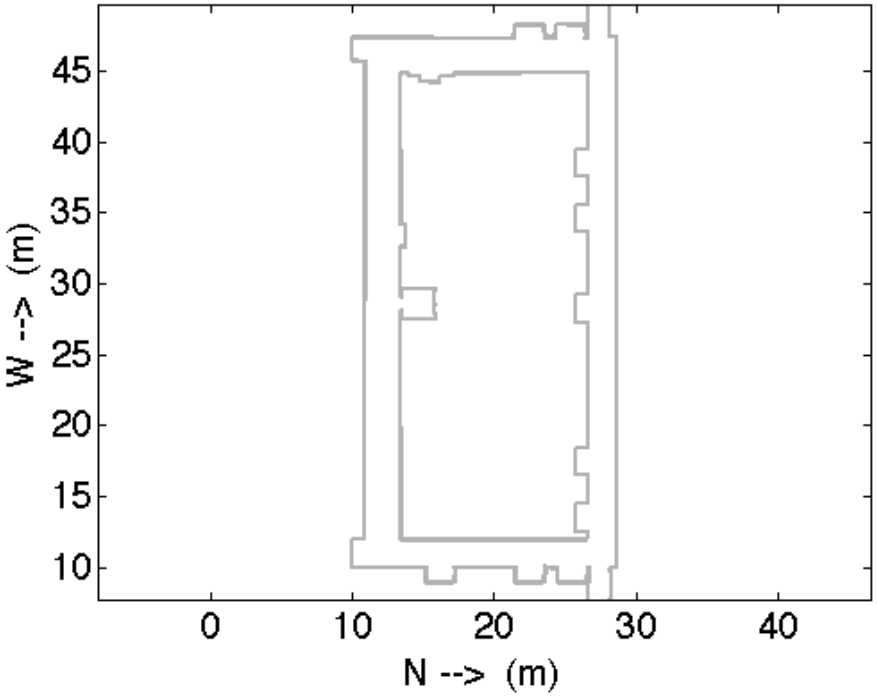

Figure 6.1: Simulated Environment

second floor hallway was then measured by hand to obtain a map of the environment as seen in Figure 6.1.

Note that this map is not simply just two rectangles to bound the robot, but encompasses a significant amount of detail such as the thickness of the door jams as seen by the zoomed in area in Figure 6.2, which will add realistic affects such as clutter for the lidar scans.

6.1.2 Generating Robot Path

The robot path shown in Figure 6.3, is constructed as a series of zig-zag path through the hallways. This path is selected as opposed to one comprised of only straight lines with turns at the intersections of the hallway because the estimation of the lidar and odometry parameters require dynamics as shown in the previous analysis of relative pose calibration performed in Chapter 4. Thus, this path should provided sufficient opportunity for the parameters to be calibrated.

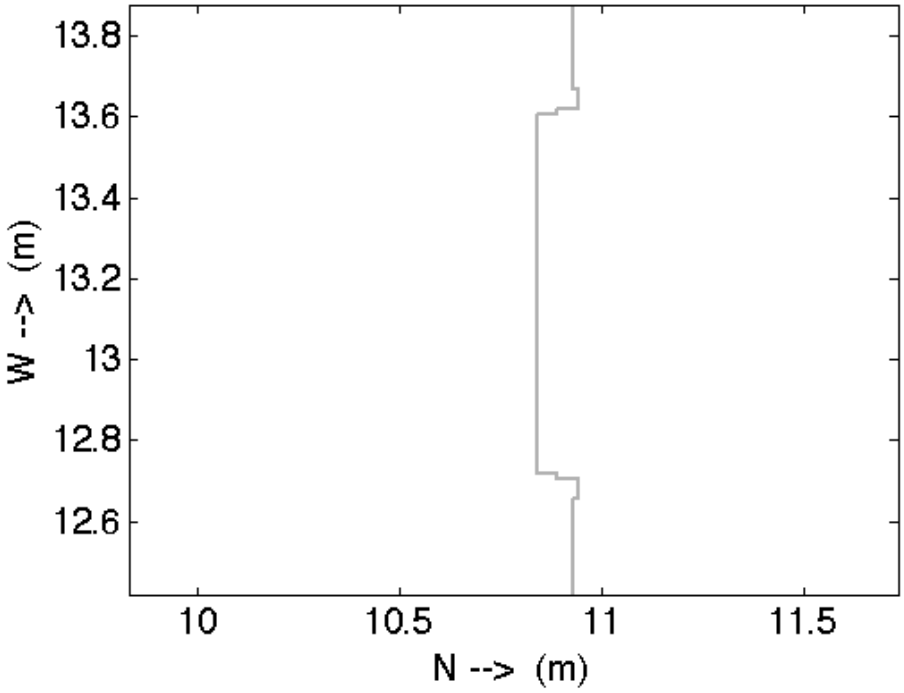

Figure 6.2: Map Detail

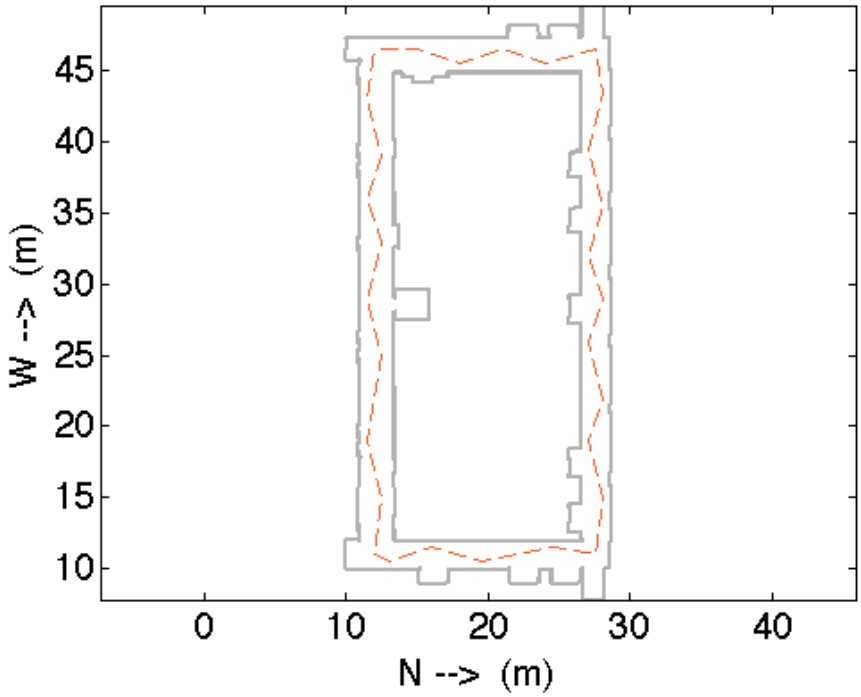

Figure 6.3: Robot Path

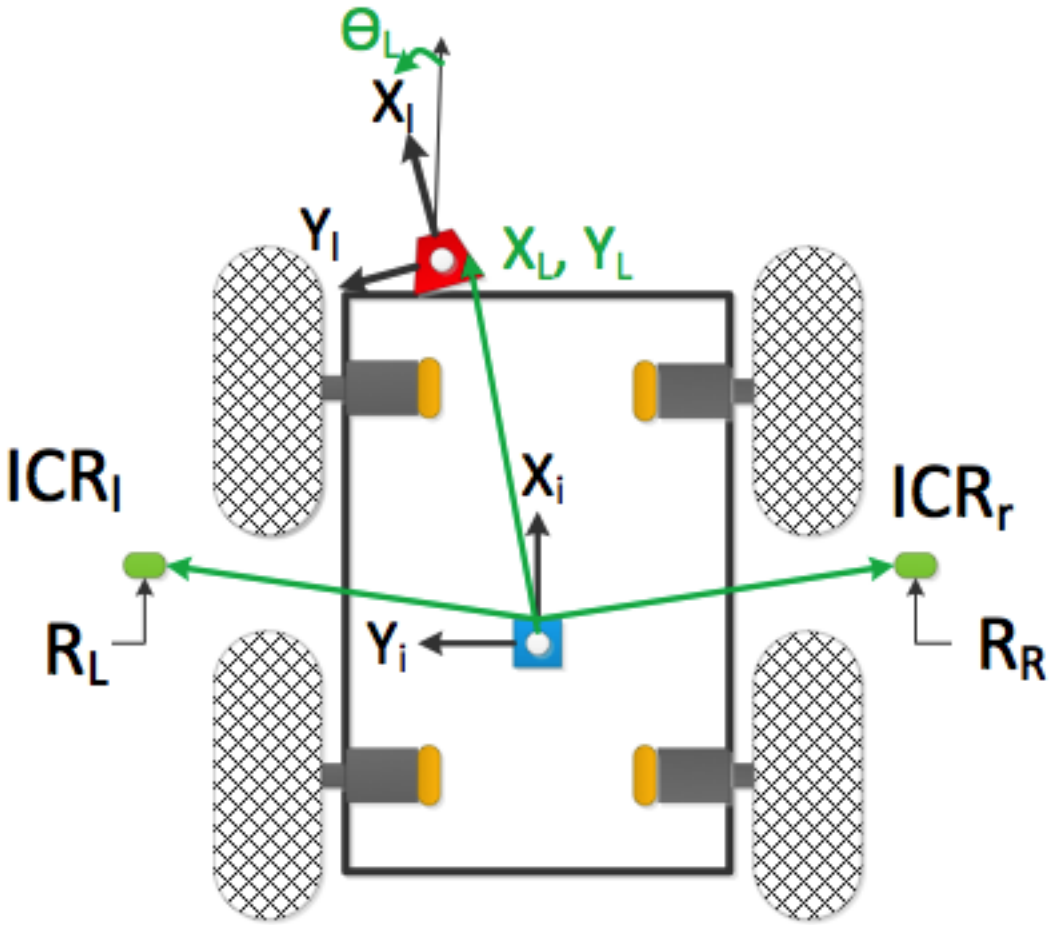

Figure 6.4: Simulated Sensor Mounting Locations

6.1.3 Generating sensor measurements

The sensors are modeled using the inherent underlying kinematic equations which represent the sensors. Therefore, to achieve representative sensor measurements, the sensors are modeled as being located as near as possible to their actual location. The location of the simulated sensor mounting locations are seen in Figure 6.4. Thus using the mounting locations, and the previously generated map and path, the individual sensor measurements can now be generated. The exact amount of offset and noise levels for the specific sensors are outlined in their respective sections.

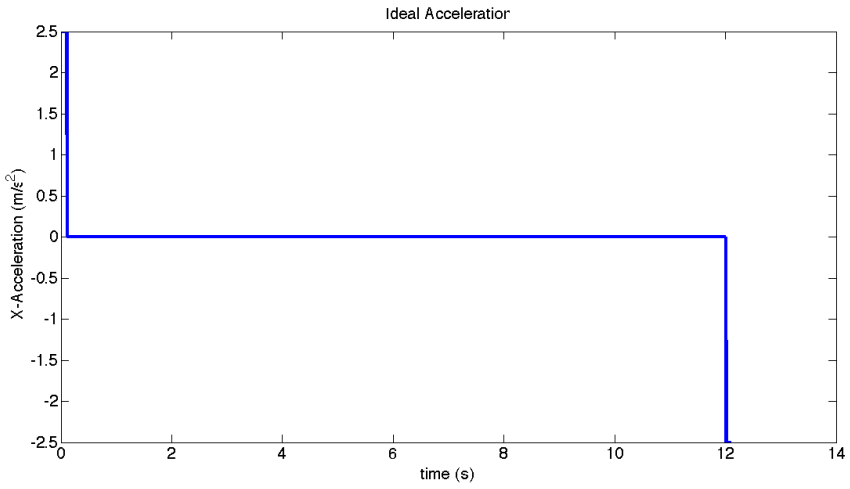

Figure 6.5: Ideal Acceleration Profile

6.1.3.1 IMU

Because all mounting locations are referenced to the IMU, the IMU is modeled as being located at the center of the vehicle. The generation of the IMU acceleration measurements begin with a velocity profile. For each straight-line portion of the robot's path the robot is modeled as ramping up to a constant velocity of $0.25m/s$ which it is allowed to achieve in $0.1s$. The robot then maintains this velocity until the end of the path subsection is reached, where the velocity is decreased until the robot stops. This acceleration profile is shown in Figure 6.5.

A period of zero velocity then occurs and the robot rotates to a desired angle. This rotation occurs in the same manner as the straight line acceleration. The robot is modeled as ramping up to a rotation rate which remains constant for a period and then ramps down. The maximum rotation rate of the robot is modeled as roughly $11.5°/\text{sec}$ which it is allowed to achieve in $0.5s$. This ideal rotation rate profile is seen in Figure 6.6. Once the desired rotation angle is achieved, another period of no motion occurs, and then straight line driving resumes and the cycle repeats.

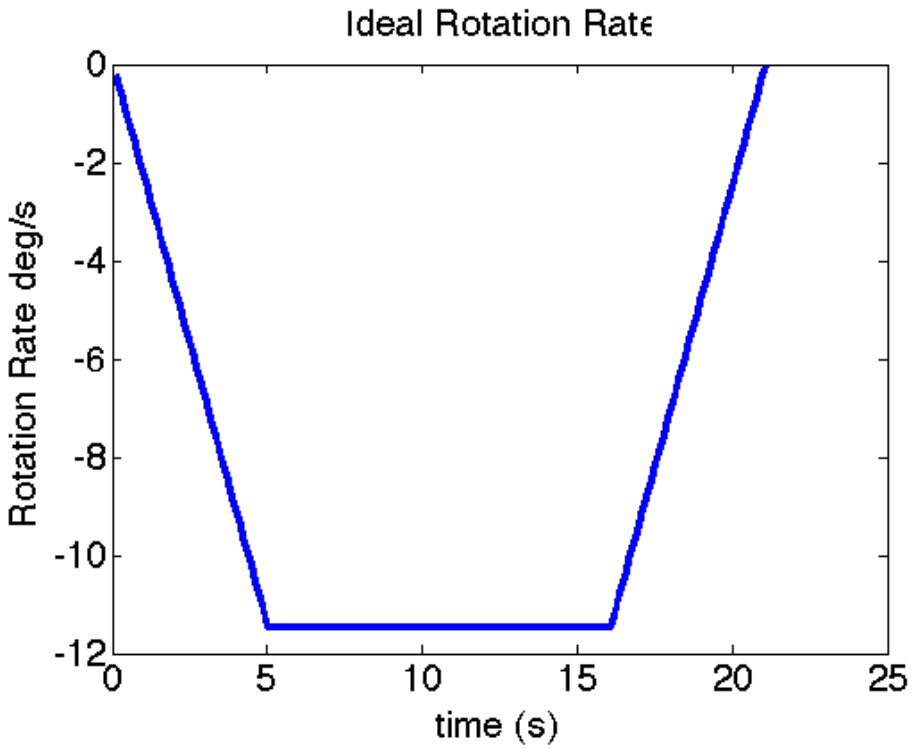

Figure 6.6: Ideal Rotation Rate

Table 6.1: IMU Noise Values

	X-Accel	Y-Accel	Z-Rotation Rate ($g_{\dot{\theta}}$)
σ_c^2	$1 \times 10^{-4}\ m/s^2$	$1 \times 10^{-4}\ m/s^2$	1.0678×10^{-4} rad$/s$
σ_w^2	$0.0277\ m/s^2$	$0.0277\ m/s^2$	1.4608×10^{-4} rad$/s$
τ	$500\ s$	$500\ s$	$1300\ s$
b	$0.1224\ m/s^2$	$-0.1433\ m/s^2$	-3.8058×10^{-4} rad$/s$
f_s (Hz)	100	100	100

Noise is then added to all axes of the IMU using the parameters given in Table 6.1. Each axis are modeled as being perturbed by a constant offset (b), white noise (w), and a moving bias (c), which are dependent on the sampling frequency (f_s) as outlined in [26].

Then allowing $\ddot{x}$ and $\ddot{y}$ to represent the ideal acceleration values, and $\dot{\theta}$ to represent the ideal rotation rate, then the output of the IMU is defined according to Equations (6.1-6.4) using notation similar to [26].

$$
\begin{aligned}
a_{\ddot{x}} &= \ddot{x} + c_{\ddot{x}} + b_{\ddot{x}} + w_x \\
a_{\ddot{y}} &= \ddot{y} + c_{\dot{y}} + b_{\ddot{y}} + w_y \\
g_{\dot{\theta}} &= \dot{\theta} + c_{\theta} + b_{\theta} + w_z
\end{aligned} \tag{6.1}
$$

The white noise for each axis is defined as follows:

$$
\begin{aligned}
E\left[w_x\right] &= \sigma_{w,x}^2 f_s \\
E\left[w_y\right] &= \sigma_{w,y}^2 f_s \\
E\left[w_z\right] &= \sigma_{w,z}^2 f_s
\end{aligned} \tag{6.2}
$$

The mowing bias is defined in the equations below and is modeled as a first order Markov process:

$$
\begin{aligned}
\dot{b}_{\ddot{x}} &= -\tfrac{1}{\tau_x} b_{\ddot{x}} + w_{c,x} \\
\dot{b}_{\ddot{y}} &= -\tfrac{1}{\tau_y} b_{\ddot{y}} + w_{c,y} \\
\dot{b}_{\dot{\theta}} &= -\tfrac{1}{\tau_x} b_{\dot{\theta}} + w_{c,z}
\end{aligned} \tag{6.3}
$$

where the noise terms for the moving bias are defined as follows

$$
\begin{aligned}
w_{c,x} &= \sqrt{\tfrac{2 f_s \sigma_{c,x}^2}{\tau_{\ddot{x}}}} v \\
w_{c,y} &= \sqrt{\tfrac{2 f_s \sigma_{c,y}^2}{\tau_{\ddot{y}}}} v \\
w_{c,z} &= \sqrt{\tfrac{2 f_s \sigma_{c,z}^2}{\tau_{\ddot{z}}}} v \\
v &\sim N\left[0, 1\right]
\end{aligned} \tag{6.4}
$$

Thus the simulated outputs for the IMU are now generated.

6.1.3.2 Odometry

The goal of the simulated odometry measurements it to emulate what a wheel encoder would see on the axle. To achieve this task, the wheel radii, and instantaneous centers of rotation for the left and right wheels must be defined as well as a desired gear ratio. The values used to simulate the odometry and be seen in Table 6.2, along with their corresponding

Table 6.2: Simulated Odometry Values

x_{ICR}	$0\ m$
y_{ICR_l}	$0.4\ m$
y_{ICR_r}	$-0.3\ m$
R_l	$0.19\ m$
R_r	$0.20\ m$
tpr	44400
f_s	$50\ Hz$
σ_d	$0.03\ m$

locations on the robot as shown previously in Figure 6.4. Recall from Section 3.5 that the ICR values effectively model the two wheels on the robot as a single unit as well as account for wheel slip. Additionally, recall that tpr is the ticks per revolution and accounts for the gear ratio between the wheel and the encoder.

To generate the correct number of tics for each wheel encoder, the true change in motion is calculated from the ideal IMU outputs. The distance the encoder travels during straight-line driving is simply the same as the distance traveled by the robot as seen in Equation (6.5),

$$\text{dist}_L(k) = \text{dist}_R(k) = \left\| \begin{array}{cc} dX(k) & dY(k) \end{array} \right\| \tag{6.5}$$

and the distance traveled during the robot rotation is the arc length the wheel makes as seen in Equation (6.6).

$$\begin{aligned} \text{dist}_L(k) &= -\text{sign}\left(d\Theta\left(k\right)\right) \left| \left\| \begin{array}{cc} x_{ICR} & y_{ICR_l} \end{array} \right\| d\Theta(k) \right| \\ \text{dist}_R(k) &= \text{sign}\left(d\Theta\left(k\right)\right) \left| \left\| \begin{array}{cc} x_{ICR} & y_{ICR_r} \end{array} \right\| d\Theta(k) \right| \end{aligned} \tag{6.6}$$

These distances represent the distance traveled for each leg. Where a leg - straight portion of zig-zag followed by turn. Noise is then applied to each distance measurement according to Equation (6.7) to simulate such that the overall error will grow with distance

traveled similar to [36].

$$\begin{aligned} d_L(k) &= d_L(k-1) + \text{dist}_L(k) \\ d_R(k) &= d_R(\text{k-1}) + \text{dist}_R(k) \\ w &\sim N(0, \sigma_d) \end{aligned} \tag{6.7}$$

Note that if there is a non-zero amount of rotation, then overall distance can be decremented for the appropriate wheel to denote that it was traveling in reverse.

Finally the tics for the left and right wheels are calculated according to Equation (6.8) based on the wheel radius and the number of tics per revolution. Note all values are rounded down to simulate the quantization noise of the encoders.

$$\begin{aligned} \text{tic}_L &= \text{floor}\left(\left(d_L(k)/R_l\right)\left(\text{tpr}/\left(2\pi\right)\right)\right) \\ \text{tic}_R &= \text{floor}\left(\left(d_R(k)/R_r\right)\left(\text{tpr}/\left(2\pi\right)\right)\right) \end{aligned} \tag{6.8}$$

Finally the time vector for the wheel odometry was constructed such that the left and right odometers where $f_s/2$ $(f_s = 50Hz)$ apart to roughly simulate practical differences in measurement capture times.

6.1.3.3 Lidar

Naturally, the goal of the simulated lidar is to generate accurate range and bearing measurements to any objects in the environment. To accomplish this, the lidar was simulated as being capable of measuring to a maximum range of 49.75 m and a field of view (FOV) of 209°. The reason for this filed of view is that while the emulated sensor is capable of a greater field of view, the physical dimensions of the robot limit the field of view as previously mentioned. Additionally the lidar is simulated as having an angular resolution of 0.25°. The range and angular measurements measurements are corrupted by white noise according to equations (6.9) and (6.10).

Table 6.3: Simulated Lidar Values

ρ_{max}	49.75 m
σ_ρ	0.005 m
FOV	209°
$\delta\theta$	0.25°
σ_θ	0.097°
X_L	0.5843 m
Y_L	0.01 m
Θ_L	$-5°$
f_s	25 Hz

$$\begin{aligned} \rho &= \hat{\rho} + v \\ v &\sim N(0, 0.005) \end{aligned} \tag{6.9}$$

$$\begin{aligned} \alpha &= \hat{\alpha} + v \\ v &\sim N(0, 0.097°) \end{aligned} \tag{6.10}$$

The lidar is simulated as having a mounting offset from the IMU of 0.5843 m longitudinally and 0.01 m laterally, with a rotational offset of $-5°$. These simulated values are summarized for convenience in Table 6.3.

The true range measurements are generated by placing the robot at the true pose as calculated from the ideal IMU outputs. Using the mounting angles of the lidar, a series of rays are generated over the FOV of the lidar at the set angular resolution. The map is treated as a polygon, and using the MATLAB® function *polxppoly* the distance at which the ray intersects the map is calculated. Because the map is entirely enclosed, an intersection is guaranteed. However, if the calculated distance is greater than the maximum range of the lidar, the distance is set to the maximum lidar range. Once these true ranges are calculated, they are corrupted with the simulated sensor errors.

6.2 Simulated Results

Using the simulated environment and sensors, the proposed SLACAM procedure was validated first through simulation. The initial pose of the robot was such that it started in the southwest most corner of the map seen in Figure 6.3 facing due north. The robot is initialized to this position for visual convenience of overlaying its results within the map, however it could be initialized with any pose with no alteration to final results. Additionally, the robot begins from a static position with no a-prior knowledge of the environment other than it will consist of features that can be reasonably modeled as lines. The sensor parameters which the SLACAM algorithm is attempting to estimate, vary from run to run to achieve a more complete understanding of the initial mounting error which can be tolerated. The simulated results are judged on two criteria. First, is the overall accuracy of the estimated sensor calibration parameter to that of the true sensor parameter. Secondly, because the simulated path is a complete loop such that the robot ends with the same pose as which it began, the offset in initial and final pose are assessed as a measure of the performance of the SLAM portion of the SLACAM algorithm A qualitative assessment of the performance of the algorithm can be seen via the map generated during the SLACAM procedure.

One representative resulting example of the simulation can be seen in Figure 6.7, with simulated results tabulated in Table 6.4. Note that the map displays the true map in grey (shown for reference only) with the robot generated map in black. The green lines show landmarks that have been merged, but are separated by some distance such that they are not one continuous wall. This most often occurs from insets in the wall for doors on opposite sides of the building. The black plus signs are current landmark line endpoints. Note that despite large initial errors in the lidar mounting, the final loop closure position is within centimeters of the initial position.

The improvement to the odometry can also be seen by comparing Figures 6.8 and 6.9, which show the path the robot would have taken if using solely the initial odometry values, and the path that the robot would have taken using the final odometry values. The

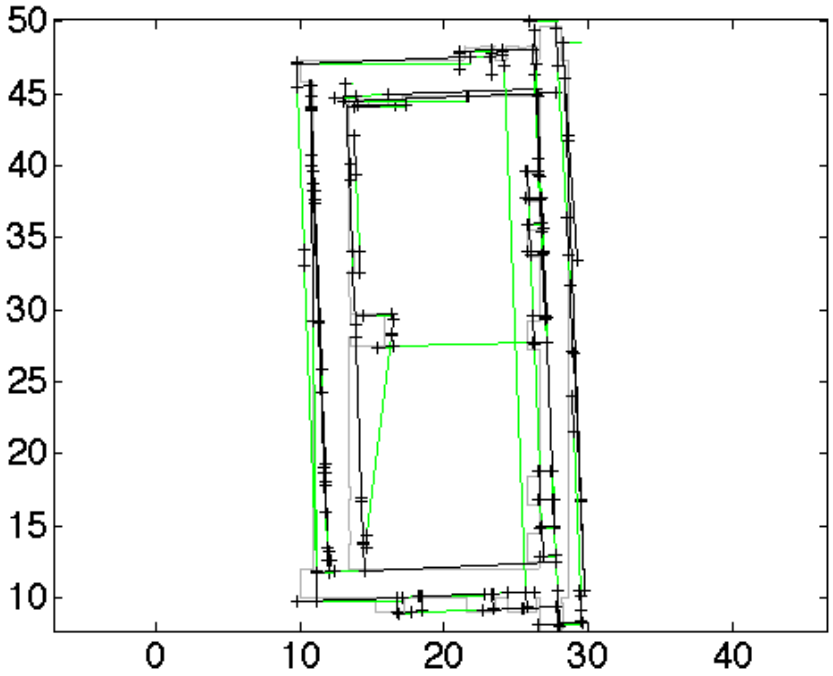

Figure 6.7: True Map in Grey with SLACAM Derived Map Overlaid in Black with Parallel Walls in Green

Table 6.4: Simulated Results

	Initial Error	Final Error
X	0	$-0.02m$
Y	0	$-0.01m$
Θ	0	$2.43°$
X_L	$1m$	$-0.20m$
Y_L	$1m$	$-0.12m$
Θ_L	$2°$	$-0.22°$
X_{ICR}	0	0.08m
Y_{ICR_L}	0	$-0.05m$
Y_{ICR_R}	$0.03m$	$0.0001m$
δ_L	$0.05m$	$-0.002m$
δ_R	0	$-0.003m$

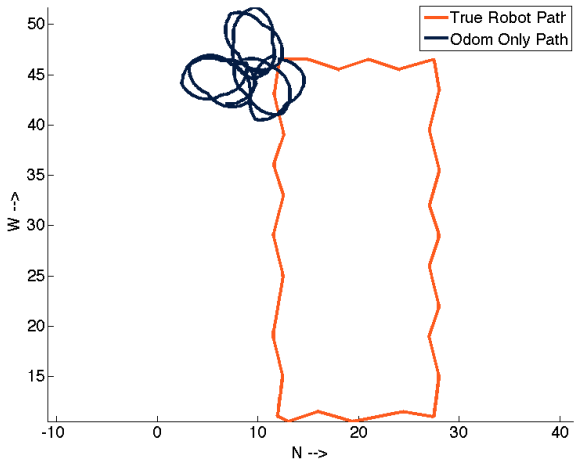

Figure 6.8: Initial Odometry Path vs True Odometry Path

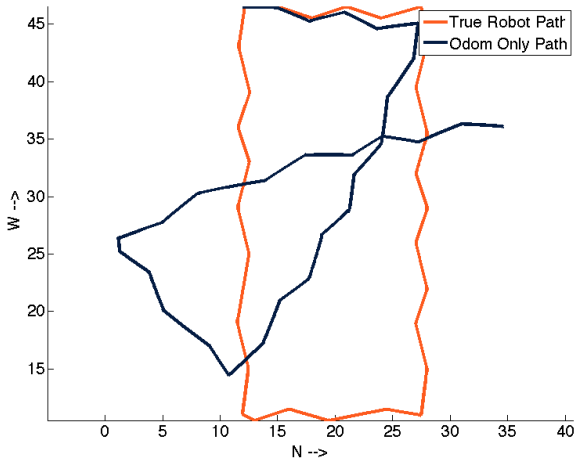

Figure 6.9: Final Odometry Path with Calibrated Results vs True Odometry Path

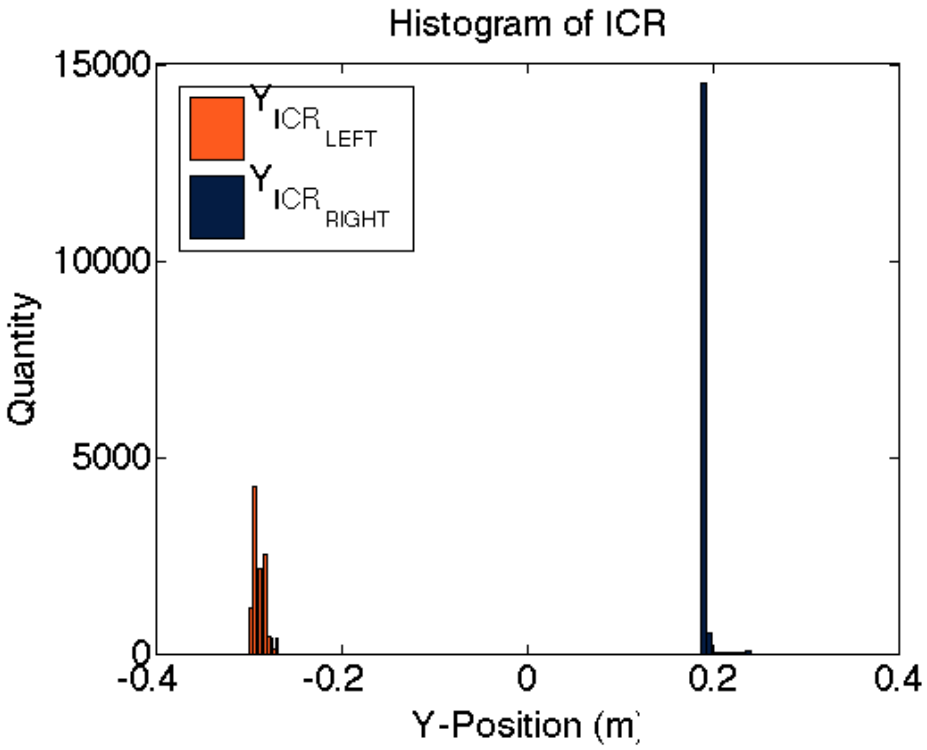

Figure 6.10: Histogram of Simulated ICR Results

final odometry values while not perfect, show a significant improvement and provide a path much closer to that of the true path taken. Note that as predicted in [93, 46, 52, 51] the ICR values should theoretically remain constant for a skid steer robot, and practically will remain bounded. The left and right ICR values are seen as a histogram in Figure 6.10. Note that they do indeed remain bounded as predicted, validating the odometry calibration methodology.

The states settling with each measurement iteration can be observed in Figure 6.11. While the odometry states settle quite quickly and remain fairly constant the lidar translation states have take slightly longer to settle once initial motion has begun. Finally the lidar mounting angle state takes the longest to settle requiring a majority of the runs data.

To determine algorithmic robustness and sensitivity to the initial mounting errors, a series of Monte-Carlo tests were preformed. Initial testing indicated primary sensitivity to the mounting angle of the lidar, as it is heavily leveraged for both calibration routines. The

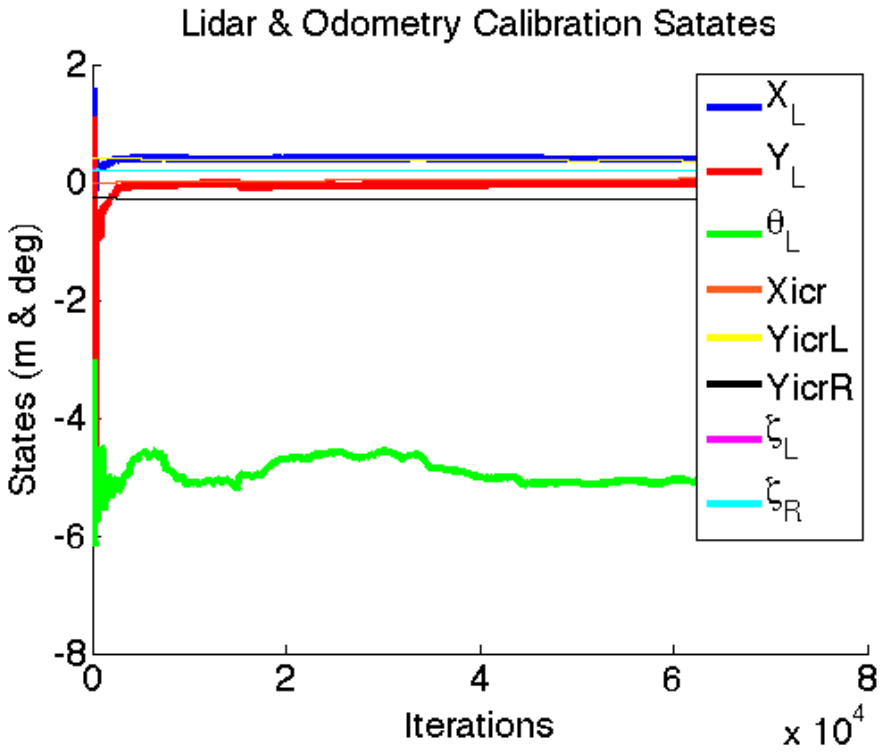

Figure 6.11: Calibration States Settling

Table 6.5: 2° Window of Lidar Mounting Error Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	1m	1m	$\pm 1°$	0m	0m	0.3m	0.05m	0m

subsections below detail initial mounting errors, and the corresponding effects on performance.

6.2.1 Lidar Angle Error - 2° Window

The first Monte-Carlo test consisted of holding all the mounting parameters static except the lidar mounting which was allowed to vary in a 2° window around the true mounting location for fifty different simulations using the parameters seen in Table 6.5. Note that while the initial odometry values are not off by more than 0.3m, the path produced is certainly not ideal as seen previously in Figure 6.8. Additionally, note that the translation error on the lidar is significant, representing a size well outside the bounds of the robot's physical size.

Table 6.6: 2° Window of Lidar Mounting Error Final Errors

	Average Error	Std Error
X	-0.018m	0.010m
Y	-0.009m	0.005m
Θ	2.265°	0.590°
X_L	-0.186m	0.049m
Y_L	-0.102m	0.028m
Θ_L	$-0.195°$	0.327°
X_{ICR}	0.073m	0.019m
Y_{ICR_L}	-0.038m	0.010m
Y_{ICR_R}	0.004m	0.003m
δ_L	-0.001m	¡0.001m
δ_R	-0.003m	0.001m

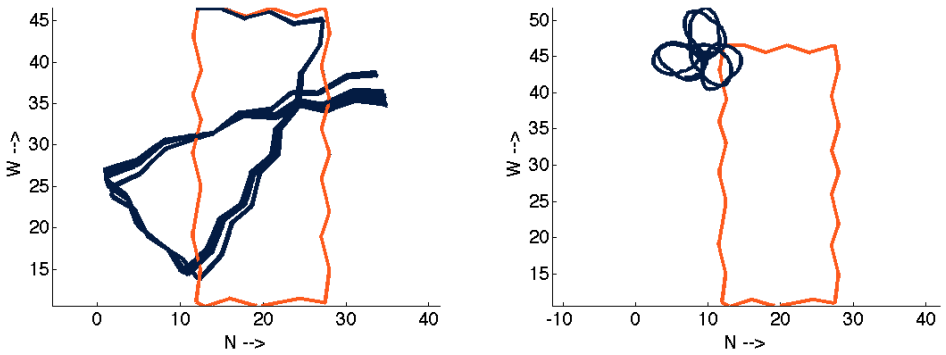

Figure 6.12: 2° Lidar Mounting Window Error Odometry Path Comparison

The results of this experiment are provided in Table 6.6. A qualitative analysis of the path errors that these odometry values constitute is provide in Figure 6.12 where the paths using the final calibrated odometry parameters are plotted as well as a comparison to the initial path errors. Note that the calibrated odometry parameters decrease the path MSE by 16.3 m as compared to the initial values.

6.2.2 Lidar Angle Error - 5° Window

This Monte-Carlo run is essentially the same as the previous analysis with the exception of the mounting error the lidar now varying over a 5° window. The initial lidar translation

Table 6.7: 5° Lidar Mounting Error Window Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	1m	1m	±2.5°	0m	0m	0.3m	0.05m	0m

Table 6.8: 5° Lidar Mounting Error Window Final Errors

	Average Error	Std Error
X	0.092m	0.310m
Y	-0.001m	0.027m
Θ	2.250°	0.550°
X_L	-0.191m	0.042m
Y_L	-0.108m	0.025m
Θ_L	−0.150°	0.700°
X_{ICR}	0.075m	0.017m
Y_{ICR_L}	-0.040m	0.009m
Y_{ICR_R}	0.003m	0.003m
δ_L	-0.002m	0.001m
δ_R	-0.003m	-0.001m

error and odometry errors remain static throughout the fifty simulations, however they are initialized with error as seen in Table 6.7.

The results of this experiment are provided in Table 6.8. A qualitative analysis of the path errors that these odometry values constitute is provide in Figure 6.13 where the paths using the final calibrated odometry parameters are plotted as well as a comparison to the initial path errors. Note that the calibrated odometry parameters decrease the path MSE by 16.2m as compared to the initial values.

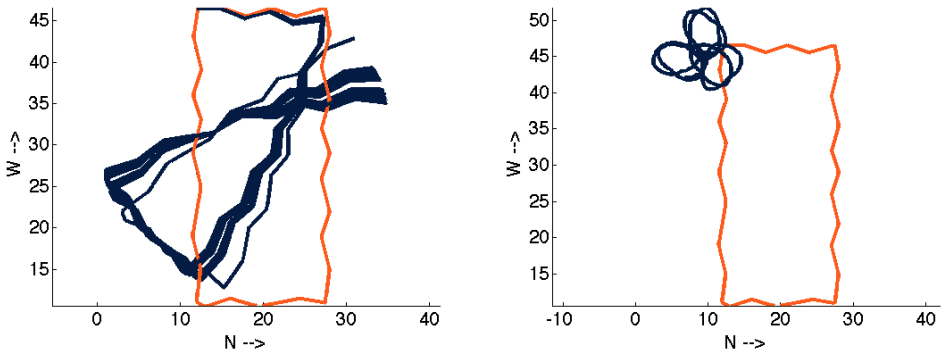

Figure 6.13: 5° Lidar Mounting Error Window Odometry Path Comparison

Table 6.9: 10° Lidar Mounting Error Window Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	1m	1m	±5°	0m	0m	0.3m	0.05m	0m

	Average Error	Std Error
X	0.664m	4.977m
Y	-0.342m	2.960m
Θ	2.317°	4.182°
X_L	-0.191m	0.034m
Y_L	-0.113m	0.061m
Θ_L	−0.291°	1.571°
X_{ICR}	0.075m	0.017m
Y_{ICR_L}	-0.040m	0.019m
Y_{ICR_R}	0.003m	0.015m
δ_L	-0.002m	0.001m
δ_R	-0.003m	0.002m

Table 6.10: 10° Window Final Errors

It appears that the lidar having an initial error lidar heading error within a 5° degree window increases the overall error negligibly compared to a 2° window. The major differences are the standard deviations in the final lidar mounting result and the final robot position in the X-axis.

6.2.3 Lidar Angle Error - 10° Window

This Monte-Carlo analysis is similar to the previous analysis where the initial odometry and lidar variables are initialized to the same wrong values at the beginning of each of the fifty runs. The lidar mounting angle is however varied uniformly over a window of 10 degrees. The initial mounting parameters used in this analysis are provided in Table 6.9.

The results of this experiment are provided in Table 6.10. A qualitative analysis of the path errors that these odometry values constitute is provide in Figure 6.14 where the paths using the final calibrated odometry parameters are plotted as well as a comparison to the initial path errors. Note that the calibrated odometry parameters decrease the path MSE by 16.566m as compared to the initial values.

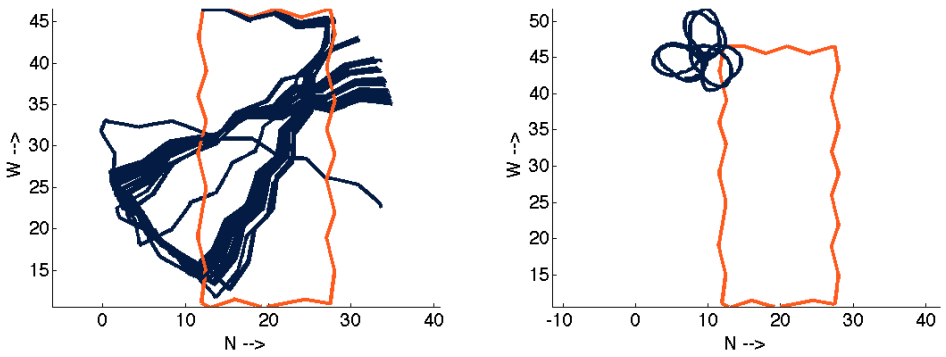

Figure 6.14: 10° Window Odometry Path Comparison

Table 6.11: Full Lidar Error Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	±0.8382/2m	±0.4572/2m	±5/2°	0m	0m	0.3m	0.05m	0m

When compared to the 5° degree window results, it can be seen that doubling the initial lidar error increases the average final position error as well as increases the standard deviation of that error. The final lidar error, however, on average remains very similar with the largest difference being the standard deviation of the lidar mounting angle. Surprisingly, the odometry average errors are nearly identical with a slight change in the standard deviation. Additionally the MSE of the path using only the newly calibrated odometry parameters is altered only slightly.

6.2.4 Full Lidar Error

Now that the effect of merely altering the initial error on the lidar mounting has been explored, the effect of the total lidar mounting error is explored. In this analysis the initial longitudinal and lateral error of the lidar is reduced from its initial significant values of 1m, and is now allowed to vary over the entire width and length of the robot, essentially signifying that the lidar mounting is merely known to be on the robot. The lidar angle is allowed to vary over a window of five degrees, while the initial odometry parameter errors remain constant throughout the fifty Monte-Carlo simulations. These initial error values can be seen in Table 6.11

Table 6.12: Full Lidar Error Final Errors

	Average Error	Std Error
X	0.130	0.356
Y	0.001	0.030
Θ	2.297°	0.391°
X_L	-0.194	0.031
Y_L	-0.110	0.019
Θ_L	$-0.151°$	0.636°
X_{ICR}	0.077m	0.012m
Y_{ICR_L}	-0.041m	0.007m
Y_{ICR_R}	0.003m	0.003m
δ_L	-0.002m	0.001m
δ_R	-0.003m	0.001m

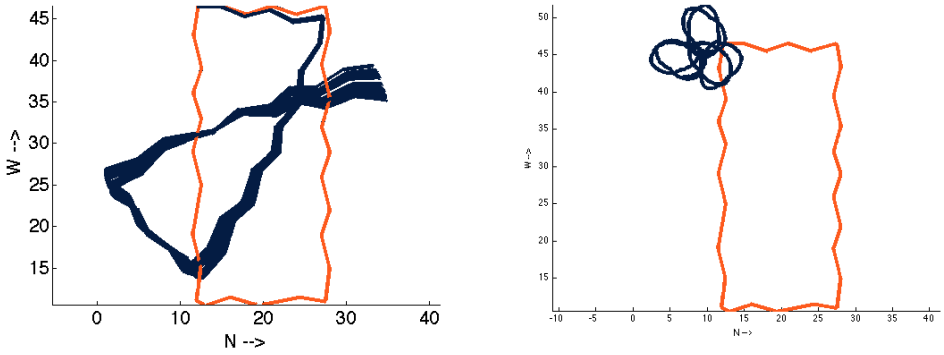

Figure 6.15: Full Lidar Error Odometry Path Comparison, Left: Calibrated Values, Right: Initial Odometry Values

The results of this experiment are provided in Table 6.12. A qualitative analysis of the path errors that these odometry values constitute is provide in Figure 6.15 where the paths using the final calibrated odometry parameters are plotted as well as a comparison to the initial path errors. Note that the calibrated odometry parameters decrease the path MSE by 17.006m as compared to the initial values.

Comparing these results to the previous analysis where only the lidar mounting angle was allowed to vary in a five degree window, it is seen the most noticeably change is an increase of roughly 0.5m in the MSE of the path when using the final odometry parameters. However, the average errors remain very similar with a slight decrease in the standard deviation in the mounting angle and final position. The reason for this slight decrease is that while the

Table 6.13: Lidar & Odometry Error Simulation Parameters

X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
$\pm$0.8382/2m	$\pm$0.4572/2m	$\pm$10/2°	$\pm$0.8382/2m	$\pm$0.4/2m	$\pm$0.3/2m	$\pm$0.19/2m	$\pm$0.2/2m

translation error for the lidar was varying a significant amount, its initial mounting error was less total error than the previous 1m / 1m of error.

6.2.5 Lidar & Odometry Error

Again fifty Monte-Carlo tests where performed such that the initial mounting error of all sensor calibration parameters varied randomly over the range denoted in Table 6.13. Note that the lidar translation mounting errors are the same as the previous Monte-Carlo analysis, however the odometry mounting parameters are now allowed to vary over the length of the robot and a significant amount in the lateral direction - nearly the width of the robot. Additionally, the initial error on the wheel radii is allowed to vary over a range the size of the wheel.

The results of this Monte-Carlo analysis can be seen in Table 6.14. When comparing these result to previous Monte-Carlo analysis which involved only lidar error, it is observed that the final position error does increase on average in the X-direction. However on average there is a decrease in the final yaw error. Note that the standard deviation of the final position does increase. The average lidar error improves a small amount due to the odometry and lidar on average being in greater initial agreement. The final odometry errors however remain very similar numerically to the previous results with the largest improvement being the X_{ICR} value. The average MSE path improvement when comparing the initial to final odometry values is 20.12m. This improvement can be seen qualitatively in Figure 6.16 which shows the paths taken using the initial and final odometry values.

These results indicate that the SLACAM algorithm is robust to large initial errors in both the lidar and odometry systems, and has the ability to estimate the true initial position even after transversing paths of roughly 100m. While the final yaw value often has more

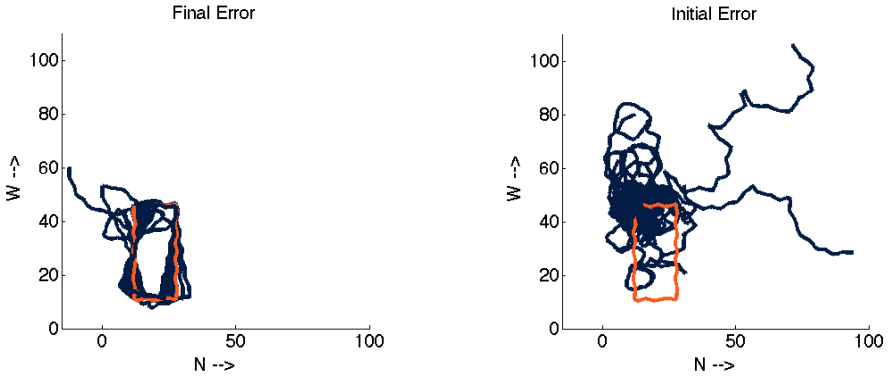

Figure 6.16: Lidar & Odom Error - Odometry Path Comparison, Left: Final Calibrated Parameters, Right: Initial Parameters

	Average Error	Std Error
X	0.76m	1.47m
Y	0.00m	2.34m
Θ	-1.46°	5.06°
X_L	0.04m	0.03m
Y_L	0.00m	0.03m
Θ_L	$-0.39°$	0.45°
X_{ICR}	-0.03	0.03m
Y_{ICR_L}	0.03	0.06m
Y_{ICR_R}	0.03	0.03m
δ_L	-0.002	0.008m
δ_R	-0.003	0.008m

Table 6.14: Lidar & Odom Error - Final Errors

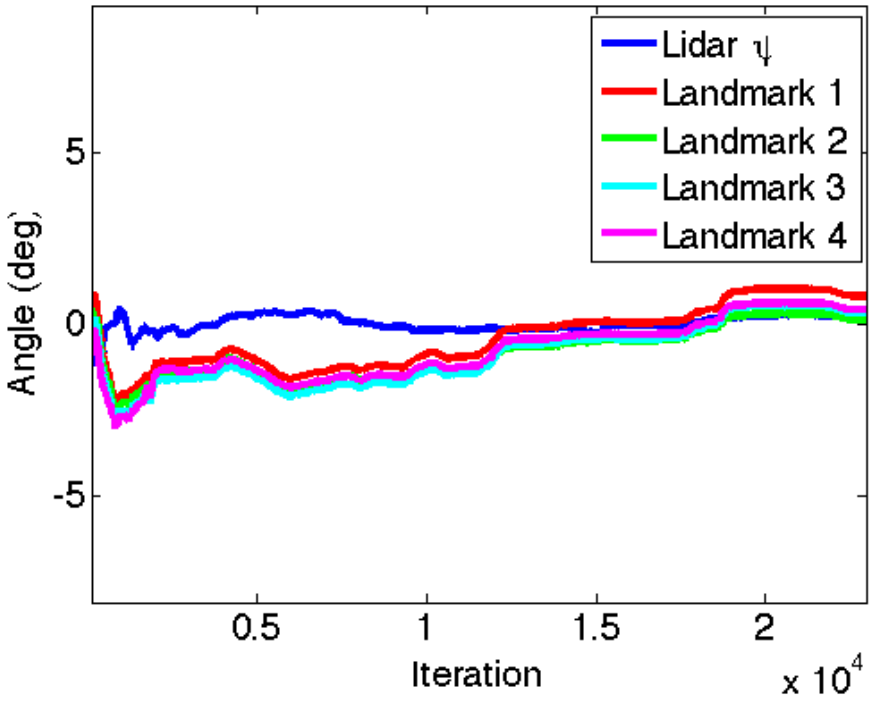

Figure 6.17: Landmark Angles Conforming to Lidar Mounting

error the than the lidar error, this is attributed to the map essentially forming to how the lidar initially perceived the world as opposed to the lidar mounting shifting to match the environment. While this is easily seen through an animation of the SLACAM process where the map can be seen to change with time, Figure 6.17 is presented as evidence of this affect where a small portion of the landmark angle estimations are plotted along with the lidar yaw angle estimation. Thus, it can be seen the the landmarks states vary much more significantly than the lidar yaw mounting indicating that the map is rotating to match the initial lidar observations.

6.3 Experimental Validation

The Experimental results seek to validate the presented SLACAM approach through the calibration of a 2D lidar and skid-steer odometry using a MEMS IMU. The experiments are conducted indoors in a structured environment. In addition to the ability of the robot

to perform calibration, the ability of the robot to map its environment as well successfully return to the same point on the map as to which it started will be investigated. This section will provide a general overview of the experimental system, hardware, and test scenario, and it conclude with how the truth values where determined and provide details on the experimentation process.

6.3.1 Overview of System

The platform on which SLACAM was performed was a modified iRobot ATRV as seen in Figure 6.18. The robot is equipped with a 2D laser scanner for general perception, which will be primarily used to map the environment and aid to accurately place the robot in its environment.

The lidar is a SICK LMS 151 laser scanner as seen in Figure 6.19 which has a maximum range of 50m. The laser scanner provided output at 25Hz with an angular resolution of 0.25°. While the lidar is advertised as having a 270° field of view, practically the lidar has a field of view of approximately 209° because the robot occludes a full 270° scan due to the mounting location.

Additionally there are encoders attached to the shaft of the motors which had a sampling frequency of 50 Hz, as well as a MEMS Crossbow-440 IMU as seen in Figure 6.20 which provided output at 100 Hz. The encoders provide both rough odometry information as well as inform the robot when it can reliably perform a ZUPT. From the motor there is an internal gear reduction and a second gear reduction occurs from the belt drive system which connect the motors to the wheels. Effectively, for each revolution of the wheel, the encoder measures 44,400 tics. Additionally note that based on this gear reduction and other physical parameters of the robot, the robot can travel at a speed of roughly $0.25m/s$ and has a turning radius of $10°/s$. Note that while the robot is capable of traveling faster, it is limited to these speeds for both safety and ease of control.

Figure 6.18: Experimental Platform

Figure 6.19: Lidar

Figure 6.20: Crossbow IMU

The robot is capable of being controlled via wireless joystick to maneuver the robot through the test environment. Additionally, the robot houses a computer for data logging and interpreting joystick commands. No additional computation or pre-filtering occurs on this computer during the gathering of experimental data.

6.3.2 Determining Truth Values

In order to have measurements to compare the experimental results, a series of offline calibrations where performed. The calibration values for the lidar where found through a series of hand measurements to determine the lidar was mounting parameters relative to the IMU. Additionally, the odometry values where found through a process similar to that of [6]. However, instead of driving in a $4m$ square grid, a series of down and back maneuvers were performed and the known distance traveled was used to determine the odometry parameters. Additionally, to mimic this calibration technique while not corrupting the results by using

Table 6.15: True Sensor Mounting Parameters

X_L	$0.58m$
Y_L	$-0.03m$
Θ_L	0°
X_{ICR}	0
Y_{ICR_L}	$0.48m$
Y_{ICR_R}	-0.48
δ_L	$0.18m$
δ_R	$0.18m$

onboard sensor measurements, the robot was driven under human power, and modeled as an ideal differential drive robot to obtain reasonable wheel odometry and wheel separation parameters. The resulting calibration parameters can be seen in Table 6.15.

6.3.3 Experimental Test Configuration

The experimental process was one that closely mimicked that of the simulated data, such that the robot was initially placed facing north in the southwest most corner of the second floor of the of the Advanced Research building in the Shelby center. The robot was then driven via the joystick in a clockwise path through the entirety of the floor in a series of zig-zag paths similar to that of Figure 6.3. Before the run began, tape was applied to the floor where the center of the robot wheels were located as seen in Figure 6.21. The robot was then maneuvered as closely as possible back to this same location such that the initial final pose of the robot would be the same. Sensor data was collected via the onboard computer and was processed offline.

6.4 Experimental Results

The results shown are from a series of tests in which the lidar and odometry calibration parameters are found relative to an IMU while the robot is also using those sensors to map and localize itself within the environment.

Figure 6.21: Marking Robot Pose

Table 6.16: Initial & Final Errors

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	0.004m	0.005m	0.76°	-0.02m	0.02m	-0.03m	-0.002m	0.01m
Final Mounting Error	0.004m	0.005m	0.25°	-0.06m	0.02m	-0.01m	0.006m	0.004m

The analysis procedure used draws heavily from both the SLAM, auto-calibration, and SLACAM literature. Lidar values were judged on the basis of their agreement with the hand calibrated values as found previously. The initial lidar mounting errors will be varied over a minimum range of at least 0.10m and 1° as seen in [42, 34]. Additionally the odometry values will be compared to both previously found odometry parameters as well as by determining the performance increase in using the calibrated odometry values by comparing the odometry only paths similar and the ability to complete a full loop similar to that of [6, 47]. Similar to the simulated results, the ability of the robot to accurately construct a map of the environment and localize itself within that environment will be assessed on its ability to determine that it indeed ended the experiment where it began, as well as create a representative map of the environment as in [75]. Finally a series of 50 Monte-Carlo simulations of varying initial mounting locations will be run to assess the accuracy and robustness of the SLACAM method by analyzing both the agreement to the laboratory calibrated parameters as well as the tightness of the standard deviation on those parameters as seen in [45, 12]

One representative dataset can be seen below where the generated map is shown in Figure 6.22, and the initial and final parameters are provided in Table 6.16. In the map the green lines show walls that are considered to be part of the same line but are separated by some additional space. The black '+' signs denote intersections of lines, and the red '+' signs denote historical line endings. Additionally note that the odometry values show an improvement of nearly 32m from the path error as calculated from using the initial and final values as seen in Figure 6.23.

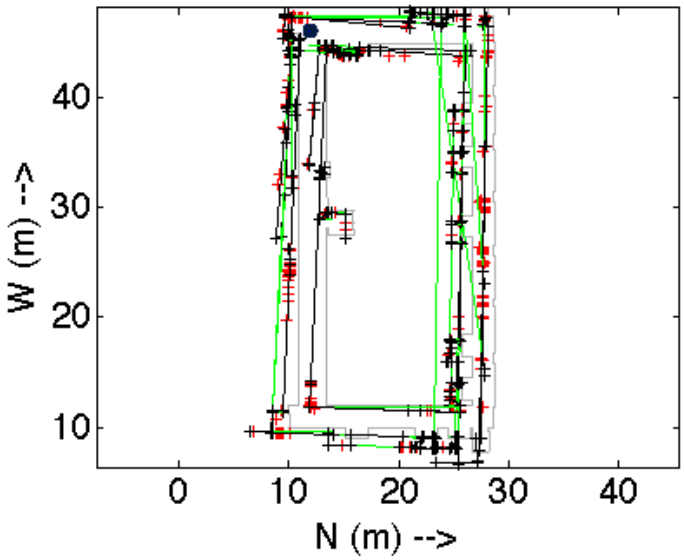

Figure 6.22: True Map in Grey with SLACAM Derived Map Overlaid in Black with Parallel Walls in Green

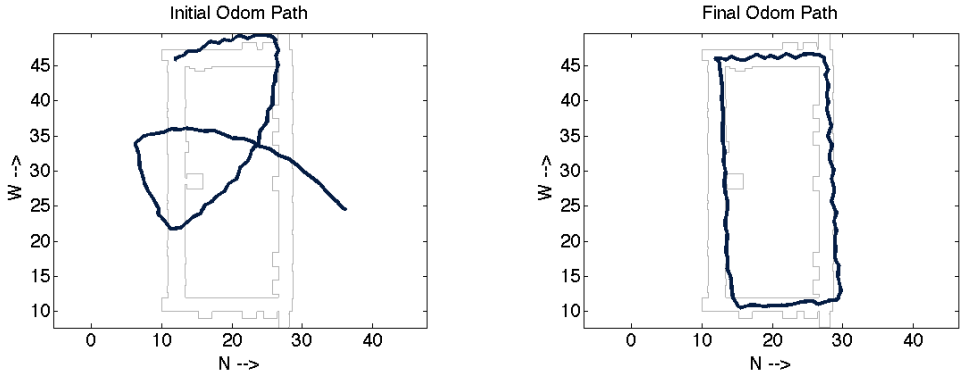

Figure 6.23: Left: Path Using Initial Odometry Parameters Right: Path Using Final Odometry Parameters

Table 6.17: 2° Window Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting	0.5842m	0.1m	$\pm 2°/2$	0m	0.4833m	-0.4833m	0.1837m	0.1837m

6.4.1 Lidar Only Error 2° Alignment

This initial test seeks to mimic the analysis performed in [45] where 2D Lidar being calibrated is provided with an initial value that is allowed to vary uniformly over a 2° window of error in the horizontal direction and all other parameters remain unchanged. The validity of the result is analyzed on the reasonableness of the mean final answer, and its standard deviation. However, instead of analyzing the results using a series of 20 different position, a Monte Carlo analysis is performed using 50 different lidar mounting locations as performed in both the simulation results as well as [12]. The initial values provided are such that the lidar translational values remain unchanged throughout the simulation and are initialized with the hand measured values, the same is true for all odometry parameters. However, the initial lidar heading mounting angle is allowed to vary over a windows of 2°. The calibration parameters used are outlined in Table 6.17.

The result of this Monte-Carlo simulation is provided in Table 6.18. To determine the accuracy of the SLAM portion of the SLACAM result, the average error between the starting and ending location is compared since the robot started and ended at the same location, the initial value of which is of no consequence as it is chosen arbitrarily by the operator. From these results it can be seen that the SLAM portion of the SLACAM solution performs quite well, with the largest error being in the $'X'$ direction (0.54m) which is proportional to the length of the robot, and the final yaw angle ($-1.8°$) and $'Y'$ error (0.003m) is to within the accuracy that the robot can be successfully driven to the same location. Analyzing the consistence of this of the SLAM portion of the SLACAM solution is achieved by analyzing the standard deviation. From this, it is seen that the algorithm is very consistent in returning to the same position. The larger standard deviation in the attitude occurs because often the algorithm ends up shifting the map to the lidar instead of the lidar to the map.

Table 6.18: 2° Lidar Window - Final Errors

	Average Final State	Average Error	Std Error
X	N/A	0.5367m	0.0012m
Y	N/A	0.0031m	0.0005 m
Θ	N/A	-1.7959°	0.5917°
X_L	0.5845m	0.0003m	2.5636e-7m
Y_L	0.1002m	0.0002m	3.5169e-6 m
Θ_L	0.6015°	0.6015°	0.0009°
X_{ICR}	-0.0566m	-0.0566m	3.2227e-5m
Y_{ICR_L}	0.5371m	0.0538m	2.6436e-4m
Y_{ICR_R}	-0.4567m	0.0266m	3.3863e-4m
δ_L	0.1890m	0.0053m	4.529e-7m
δ_R	0.1875m	0.0038m	4.739e-7m

Continuing the analysis of the ability of the SLACAM algorithm to accurately calibrate the lidar using Table 6.18, it appears that the lidar translation values show strong agreement with the hand measured values with negligible standard deviation between measurements. However, the final angle as determined by SLACAM shows over half a degree of error (0.6°) when compared to the hand measurements. However, because the initial yaw error is quite challenging to measure accurately and because of the minimal standard deviation, there is a high likelihood that the SLACAM algorithm is out performing the hand measurements. As a point of comparison, the standard deviation for a similar lidar yaw error amount seen in [45] was 0.3°.

Before comparing the odometry related results from the SLACAM algorithm, note that using the initial values for the odometry parameters as determined through initial laboratory testing provides the path seen in Figure 6.24, if no other sensors aid in determining the robot position. While these values provide a fairly accurate description of the true path taken, they are not as accurate as one would desire which if left un-calibrated would inject error into a SLAM only solution.

Quantitatively, the ICR values differ from the laboratory values by approximately 5cm, however this difference is very consistent as seen by analyzing the standard deviations. The wheel radii values however are accurate to within less than a centimeter of the laboratory

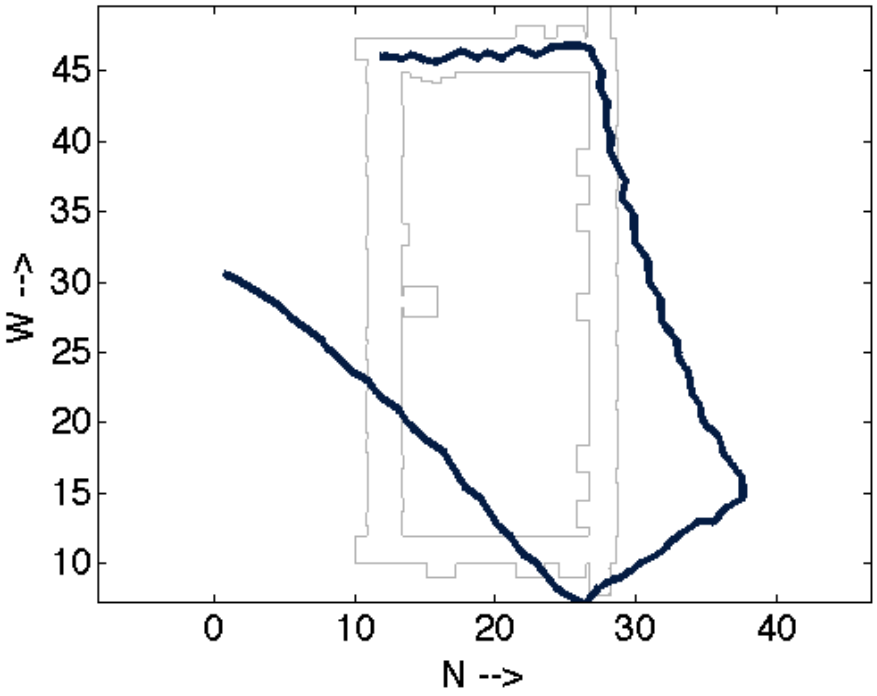

Figure 6.24: Robot Path Using Only Initial Odometry Values

calibrated parameters and remain very consistent from run to run. The qualitative performance increase in using the SLACAM odometry values is seen in Figure 6.25, where all fifty final odometry only paths are shown. The improvement of the odometry only solution to return to the initial location using the SLACAM values over the initial values is obvious.

Further analysis of the odometry path improvement is seen in Table 6.19, where the initial odom parameters yielded a final error of 18.8m while the SLACAM calibrated parameters had a final path error of 2.6m. While the SLACAM results have some error compared to the laboratory calibrated results, the SLACAM result clearly appear to perform better than the initial laboratory parameters, in addition to being very consistent. This initial Monte-Carlo analysis shows that the SLACAM algorithm performs well in both navigating and calibrating small initial lidar angle mounting and odometry errors.

A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.26, and the final pose errors are shown in Table 6.20.

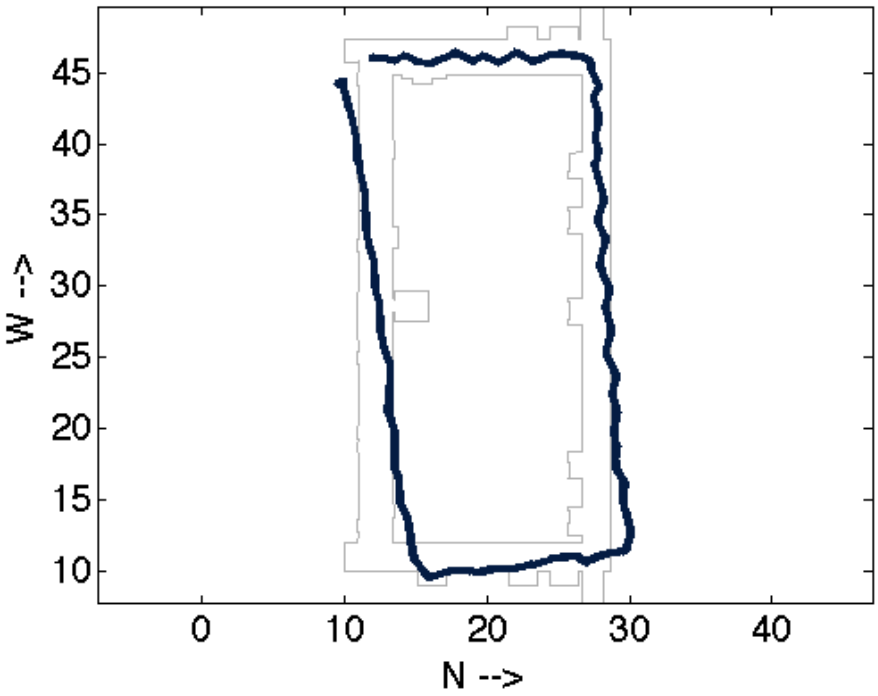

Figure 6.25: Robot Path Using Final Odometry Values

Table 6.19: 2° Window Odometry Values

	Error
Initial Mounting Path Error	18.8329m
Final Mounting Path Error	2.6351m
Odom Path Error Improvement	16.20m
Std of Odom Path Error Improvement	0.02m

Table 6.20: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5456 m	-0.0117 m	−1.6685°

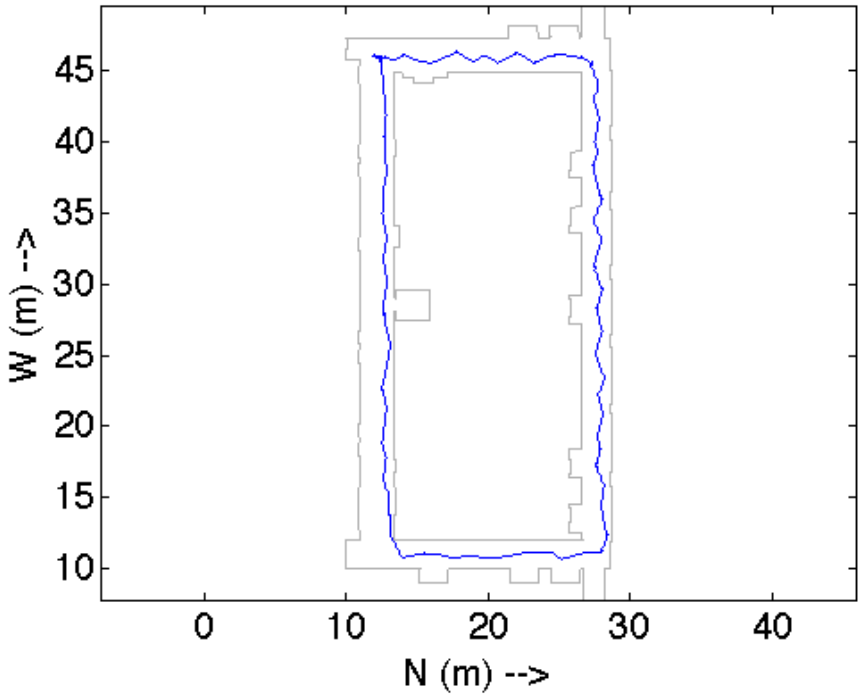

Figure 6.26: SLAM Only Solution

Table 6.21: 4° Window Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting	0.5842m	0.1m	$\pm 4°/2$	0m	0.4833m	-0.4833m	0.1837m	0.1837m

6.4.2 Lidar only Error 4° window

This Monte-Carlo analysis parallels that of Section 6.4.1, however these 50 Monte-Carlo simulations increase the range which the initial lidar mounting is allowed to vary from 2° to 4° to analyze both full system performance and determine what effect this change has over the 2° window analysis. As such, all values other than the lidar mounting angle are initialized with their laboratory calibrated values provided in Table 6.21.

The final states, errors, and standard deviations of the errors associated with this Monte-Carlo simulation are provided in Table 6.22. Once again, the SLAM portion of the SLACAM solution is quite accurate with the largest positioning error being proportional to the length of the vehicle. Additionally, the lidar calibration closely matches that of the hand measurements

Table 6.22: 4° Lidar Window - Final Errors

	Average Final State	Average Error	Std Error
X	N/A	0.5367m	0.0022m
Y	N/A	0.0031m	0.0010m
Θ	N/A	-1.7959°	1.1854°
X_L	0.5845m	0.0003m	8.9814e-07m
Y_L	0.1002m	0.0002m	7.7090e-06m
Θ_L	0.6027°	0.6027°	0.0031°
X_{ICR}	-0.0566m	-0.0566m	6.2102e-05m
Y_{ICR_L}	0.5372m	-0.0539	5.2434e-04m
Y_{ICR_R}	-0.4566m	0.0267	6.7373e-04m
δ_L	0.1890m	-0.0053	1.1380e-06m
δ_R	0.1875m	-0.0038	6.5674e-07m

with a 0.6° degree error in the yaw measurement, which is the most challenging to measure. Additionally, The lidar calibration values are extremely consistent. Comparing these results to those in Section 6.4.1 with the 2° window which the lidar was allowed to vary over, they are nearly identical both in terms of final value and standard deviation. The largest difference between the two values is an increase in the standard deviation of the final pose of the vehicle. This further validates that lidar mounting angle found here is more representative of the true mounting angle than the laboratory value.

As with the results in Section 6.4.1, the odometry ICR values differ from the laboratory calibrated values by nearly $5cm$, however they have very similar wheel radii. Qualitatively, the initial values, which are the laboratory calibration parameters, used in an odometry only solution can be seen in Figure 6.27.

The odometry only path using the final calibration parameters are seen in Figure 6.28. It appears that despite the 5cm error, the final odometry perform better than the laboratory calibration parameters.

Quantitatively the final calibration parameters in an odometry only solution return the robot roughly 16m closer to the initial starting position, as seen in the results given in Table 6.23.

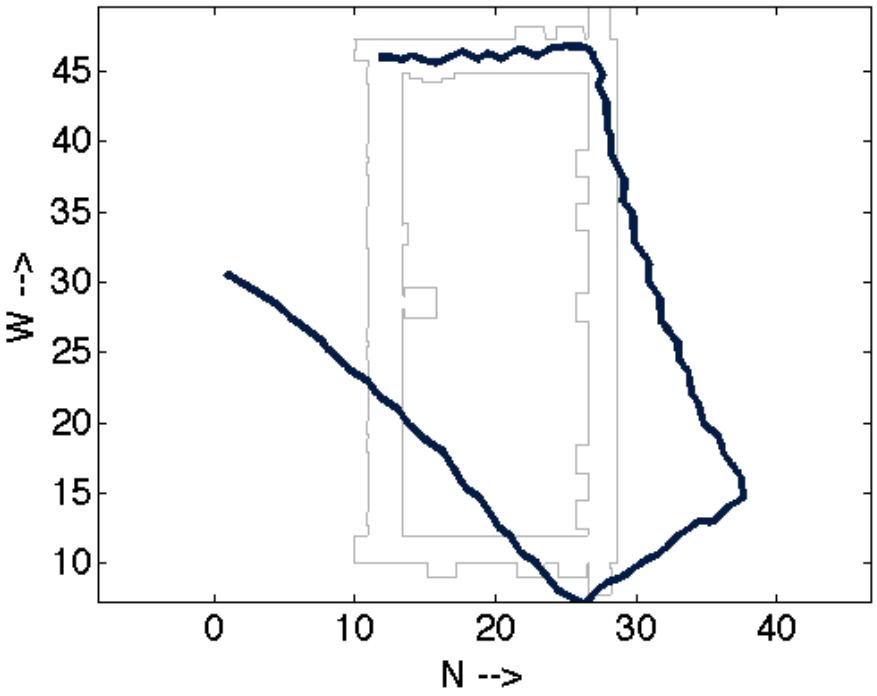

Figure 6.27: 4° Lidar Window - Initial Odom Path

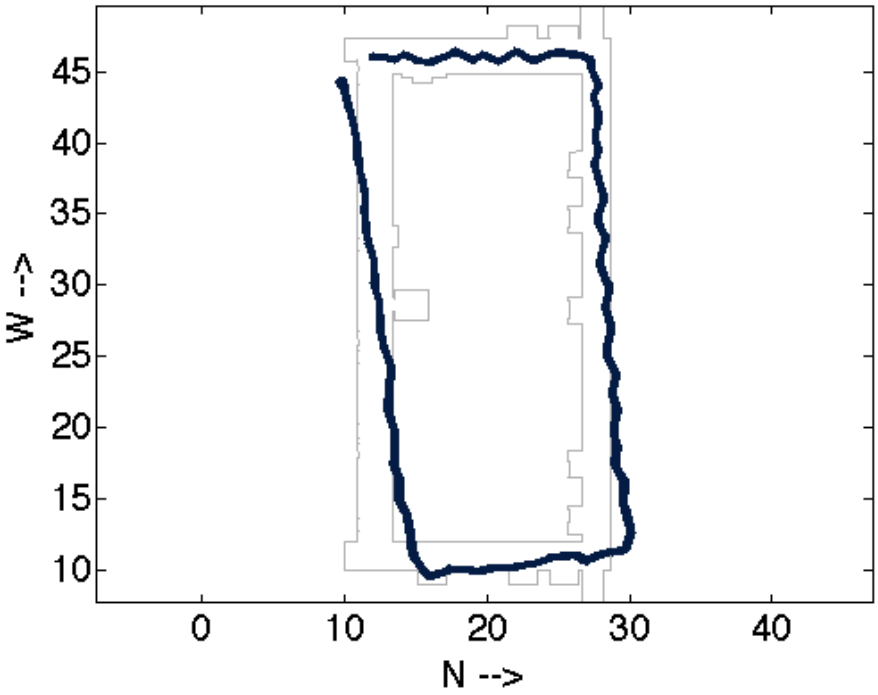

Figure 6.28: 4° Lidar Window - Final Odom Path

Table 6.23: 4° Window Odometry Values

	Error
Initial Mounting Path Error	18.8329m
Final Mounting Path Error	2.6323m
Odom Path Error Improvement	16.20m
Std of Odom Path Error Improvement	0.04m

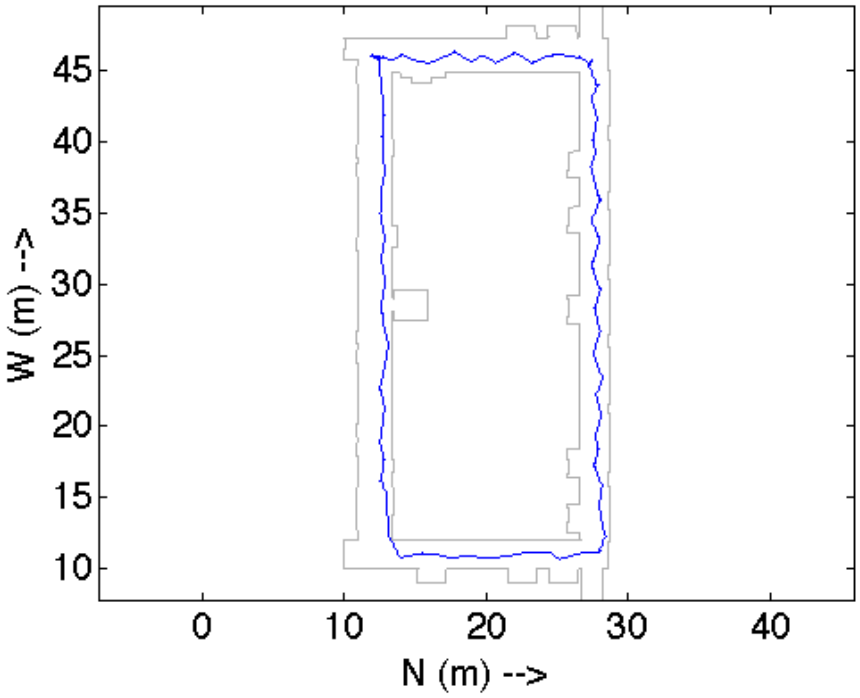

Figure 6.29: SLAM Only Solution

These results indicate that the SLACAM algorithm is robust to small errors in both the lidar mounting angle and odometry. A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.29, and the final pose errors are shown in Table 6.24.

Table 6.24: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5457 m	-0.0117 m	−1.6685°

Table 6.25: 10° Window Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting	0.5842m	0.1m	$\pm 10°/2$	0m	0.4833m	-0.4833m	0.1837m	0.1837m

Table 6.26: 10° Lidar Window - Final Errors

	Average Final State	Average Error	Std Error
X	N/A	0.5376m	0.0047m
Y	N/A	0.0035m	0.0024m
Θ	N/A	$-2.1181°$	2.9282°
X_L	0.5845m	3.0000e-04m	6.7223e-06m
Y_L	0.1002m	0.002m	7.8948e-05m
Θ_L	0.6119°	0.6119°	0.0216°
X_{ICR}	-0.0567m	-0.0567m	1.1747e-04m
Y_{ICR_L}	0.5373m	0.0540m	0.0013m
Y_{ICR_R}	-0.4564m	0.0269m	0.0017m
δ_L	0.1890m	0.0053	2.8611e-06m
δ_R	0.1875m	0.0038	1.6843e-06m

6.4.3 Lidar only Error 10° window

This Monte Carlo analysis continues to mimic that of Section 6.4.1 where all values are initialized to the laboratory calibrated values with the exception of the lidar yaw mounting, which is allowed to vary over a 10° window. Again, 50 runs are performed where the robot returns to the same initial location from which it started. The initial sensor mounting values used, are seen in Table 6.25.

The results of this Monte-Carlo analysis is seen in Table 6.26. These results indicate that even with a more sizable initial lidar mounting error, the SLACAM algorithm can still compensate from the large error and successfully return to within a robot's length of its initial location. As with 6.4.1 the final SLACAM lidar values closely match those of the laboratory measurements. The most significant difference when comparing the SLACAM results to the laboratory measurements is that of the lidar yaw angle. However, the standard deviation of this result is small, and is nearly identical to the previous Monte-Carlo analysis performed thus far giving high confidence that this is indeed a more accurate representation of the actual mounting parameters.

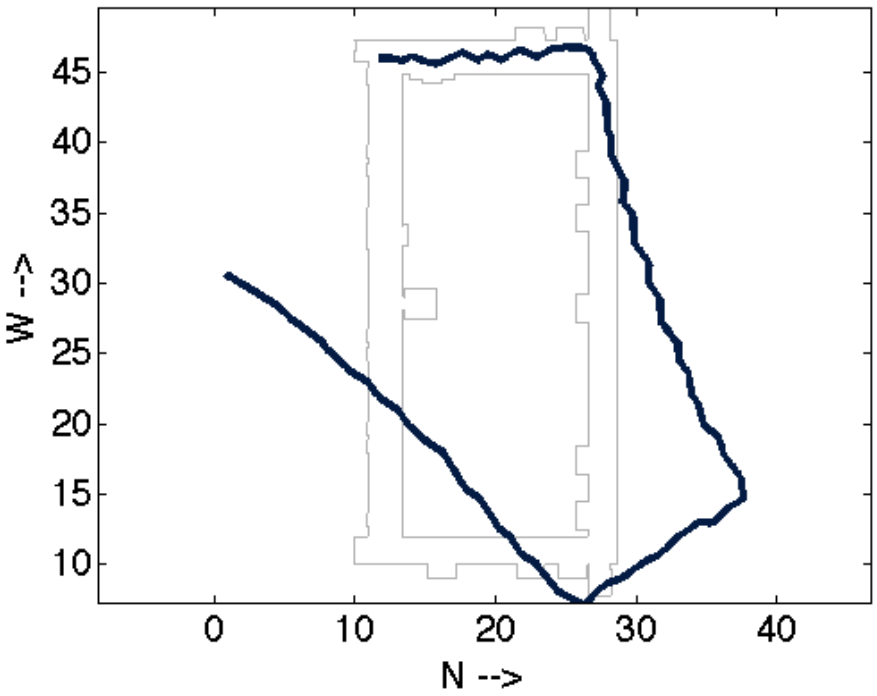

Figure 6.30: Robot Path using Initial Odometry Values

The odometry parameters found using SLACAM are again nearly 5cm off of the laboratory measurements when comparing the ICR values. This implies that these values are more likely to be correct than those found for the laboratory calibration. However, the wheel radii match very closely and all standard deviations of are very low. To assess the accuracy of the odometry calibration parameters, the initial laboratory values are used to generate an odometry only path as seen in Figure 6.30.

The odometry only path using the final calibration values as determined by SLACAM are seen in Figure 6.31. Note this figure does not show just the average final odom values, but all fifty final odometry values. The reason it appears as one path is because of the minimal variance in the calibrated parameters.

A quantitative analysis of the odometry parameters is provided in Table 6.27. Note that when comparing the odometry only path, the SLACAM solution get approximately 16m closer to the true ending position than the laboratory calibrated parameters.

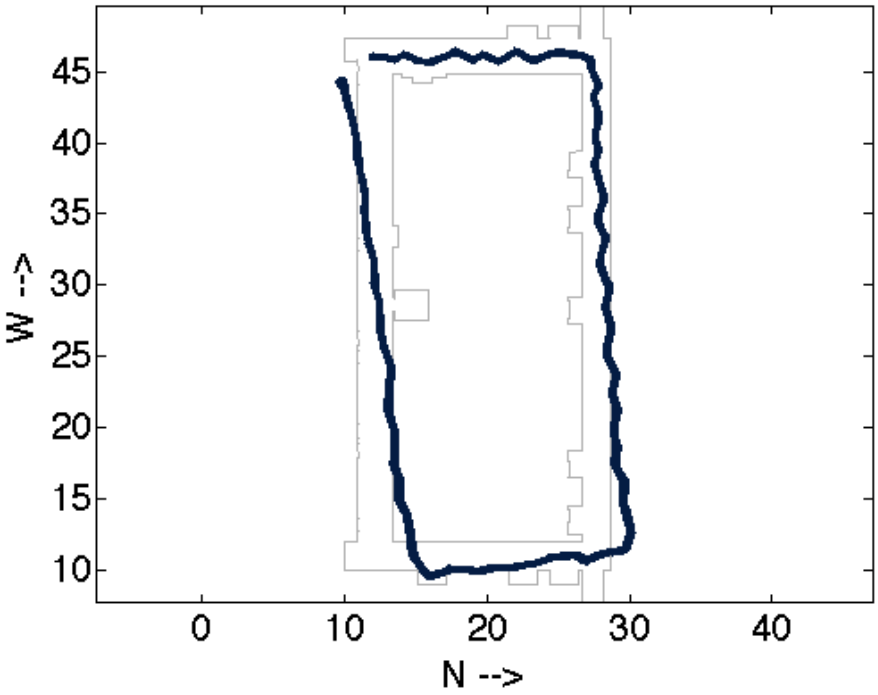

Figure 6.31: Robot Path using Final Odometry Values

Table 6.27: 10° Window Odometry Values

	Error
Initial Mounting Path Error	18.8329m
Final Mounting Path Error	2.6188m
Odom Path Error Improvement	16.21m
Std of Odom Path Error Improvement	0.08m

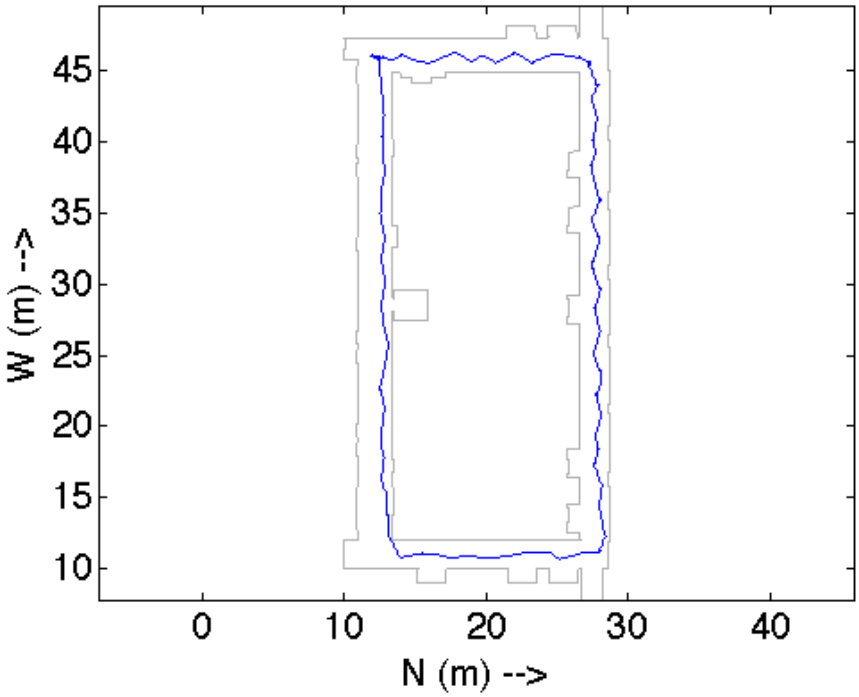

Figure 6.32: SLAM Only Solution

Table 6.28: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5457 m	-0.0116 m	$-1.6685°$

These Monte-Carlo tests indicate that the SLACAM algorithm in not sensitive to change in the initial mounting location of the lidar, and is capable of calibrating odometry from small odometry errors. A SLAM only solution was generated from the final calibration parameters. A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.32, and the final pose errors are shown in Table 6.28.

6.4.4 Lidar 10 deg window & 10 cm of Error

In this section a Monte-Carlo analysis is similar to that of Section 6.4.3 where the lidar is allowed to vary over a window of ten degrees, and the odometry values are initialized to the laboratory calibrated values. However, unlike previous simulations, this Monte-Carlo

Table 6.29: 10° & 10cm Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	$\pm 0.1m/2$	$\pm 0.1m/2$	$\pm 10°/2$	0m	0m	0m	0m	0m

Table 6.30: 10° & 10cm Simulation Parameters - Final Errors

	Average Final State	Average Error	Std Error
X	N/A	0.5365m	0.0054m
Y	N/A	0.0038m	0.0036m
Θ	N/A	$-2.1084°$	2.9180°
X_L	0.5911m	0.0069m	0.0312m
Y_L	0.0992m	0.0008m	0.0278m
Θ_L	0.6088°	0.6088°	0.1438°
X_{ICR}	-0.0606m	-0.0606m	0.0189m
Y_{ICR_L}	0.5368m	0.0535m	0.0116m
Y_{ICR_R}	-0.4568m	0.0265m	0.0116m
δ_L	0.1890m	0.0053	1.1859e-05m
δ_R	0.1875m	0.0038	9.2431e-05m

analysis also varies the lidar translation parameters over a maximum range of 10cm using the values in Table 6.29.

The results this Monte-Carlo analysis are provided in Table 6.30. Note that the final position error remain nearly identical to the previous test. Additionally, final lidar error shows no appreciable change, and the SLACAM calibration values match the laboratory translation measurements to within less than 1cm. Again the yaw value on the lidar remains nearly identical to the values found in the previous simulations. However the standard deviations across the board did increase as compared to the lidar angle only analysis performed in Section 6.4.3. Note that the standard deviation of the lidar result is comparable to that of a similar analysis performed in [45] which had slightly less initial translation error.

While the SLACAM odometry values differ by roughly 5cm from the ICR values found from the laboratory testing, the standard deviation on these measurements remain quite small. A plot of an odometry only solution is seen in Figure 6.33, which uses the initial laboratory calibration parameters.

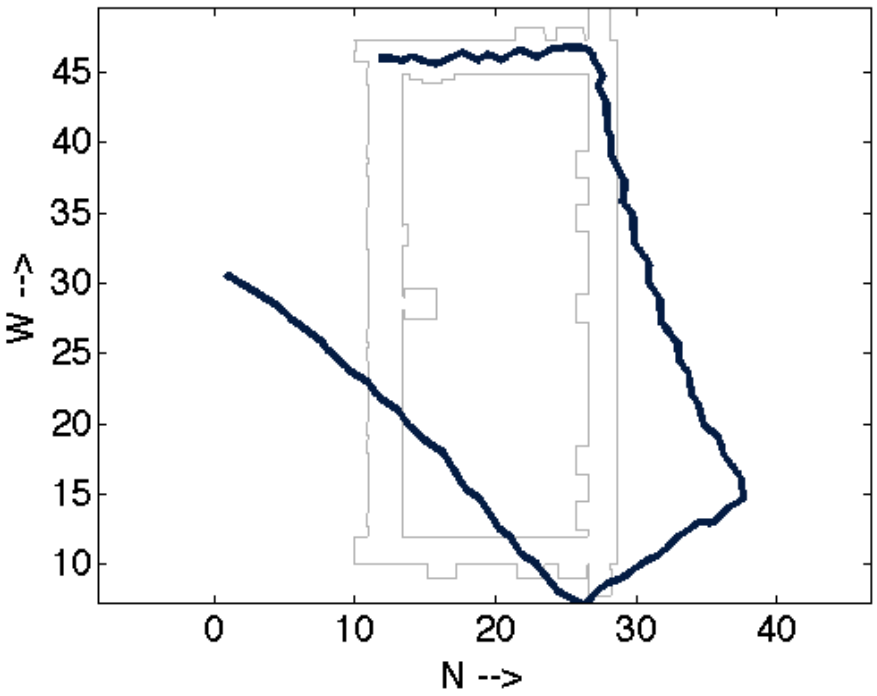

Figure 6.33: Robot Path Using Only Initial Odometry Values

Table 6.31: 10° & 10cm Simulation Parameters Odometry Values

	Error
Initial Mounting Path Error	18.8329
Final Mounting Path Error	2.5644
Odom Path Error Improvement	16.27m
Std of Odom Path Error Improvement	1.00m

A qualitative increase in the odometry only solution can be seen in Figure 6.34, where all final odometry values are plotted. While there is certainly more path variance than in 6.4.3, there is still an obvious performance increase.

The odometry performance increase is seen in Table 6.31, where the SLACAM calibrated results are on average 16m closer to the true ending position than then initial values derived from the laboratory calibration.

While this Monte-Carlo analysis shows that the SLACAM algorithm operates well even from some initial lidar error, this mounting error does have a small but noticeable affect on the odometry calibration parameters. However, no appreciable affect is seen on the ability

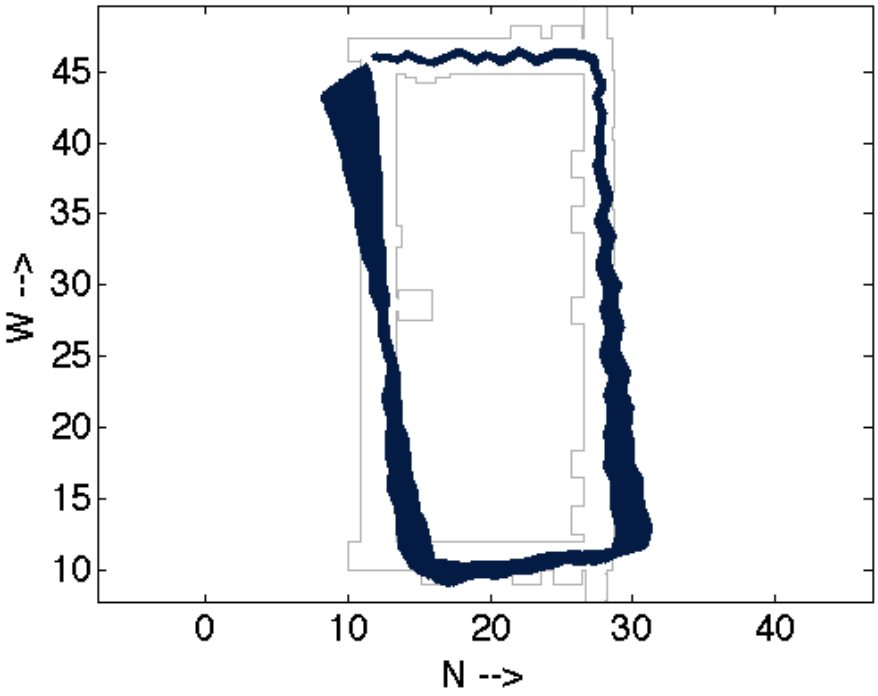

Figure 6.34: Robot Path Using Only Final Odometry Values

Table 6.32: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5451 m	-0.0114 m	$-1.6682°$

of the robot to return to its initial starting location. A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.35, and the final pose errors are shown in Table 6.32.

6.4.5 Lidar Error Ranging Over Entire Dimension of Robot

In an effort to further analyze the total system robustness, another Monte-Carlo analysis is performed similar to that of Section 6.4.4, where the fifty Monte-Carlo simulations vary all of the lidar parameters while initializing the odometry values with the laboratory measurements. However, the translation error is increased significantly such that the lidar is varied over a range comparable to that of the full length and width of the robot. This is equivalent to installing a lidar on the platform and having no concept where it was actually

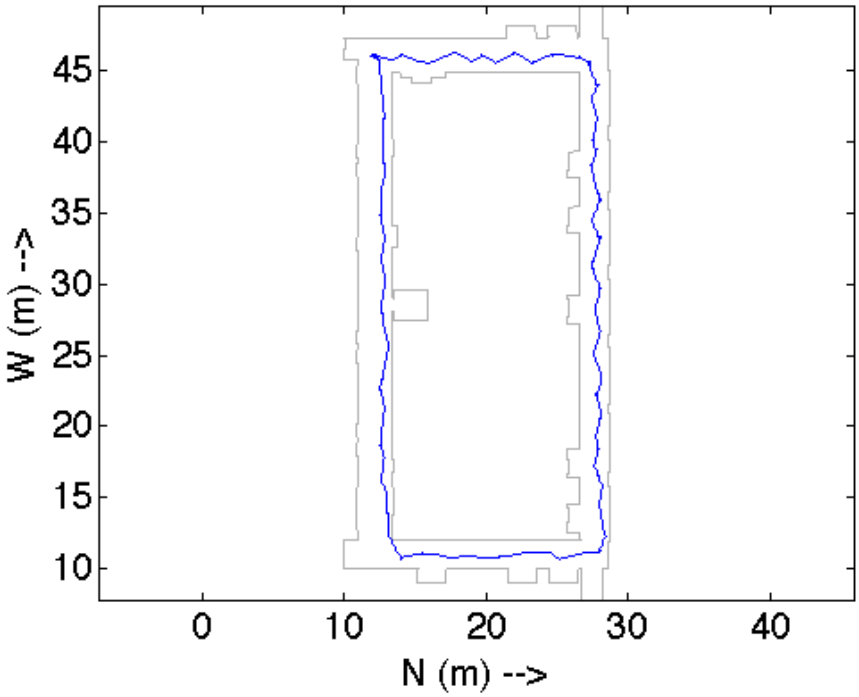

Figure 6.35: SLAM Only Solution

Table 6.33: Full Lidar Error Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	$\pm\frac{0.8382m}{2}$	$\pm\frac{0.4572m}{2}$	$\pm 10°/2$	0m	0m	0m	0m	0m

located, except knowing it was generally facing forwards. These initial simulation parameters are given in Table 6.33.

The final results of this Monte-Carlo analysis are provided in Table 6.34. The most noteworthy change in the average error between these results and those of 6.4.4 is that of the lidar yaw measurement which increase by 0.15°. While this is not a sizable amount, it is worth noting that there is a correspondence between the size of the initial translational error and that it can impact the final SLACAM lidar yaw value. Additionally, as expected, there is also an increase in the standard deviation of the lidar translation parameters.

The odometry values also had a small increase in error from the previous analysis in Section 6.4.4, where a change of roughly 1cm in X_{ICR} was experienced. To achieve a

	Average Final State	Average Error	Std Error
X	N/A	0.5409m	0.0191m
Y	N/A	0.0086m	0.0124m
Θ	N/A	$-2.0936°$	$2.9361°$
X_L	0.6121m	0.0279m	0.1308m
Y_L	0.0976m	-0.0024m	0.0636m
Θ_L	$0.7574°$	$0.7574°$	$0.4212°$
X_{ICR}	-0.0734m	-0.0734m	0.0793m
Y_{ICR_L}	0.5355m	0.0522m	0.0266m
Y_{ICR_R}	-0.4570m	0.0263m	0.0264m
δ_L	0.1890m	0.0053	6.1407e-05m
δ_R	0.1875m	0.0038	2.1018e-04m

Table 6.34: Full Lidar Error Simulation Parameters - Final Errors

Table 6.35: Full Lidar Error Simulation Parameters Odometry Values

	Error
Initial Mounting Path Error	18.8329m
Final Mounting Path Error	2.04m
Odom Path Error Improvement	16.14m
Std of Odom Path Error Improvement	2.6932m

qualitative sense for improvement, the laboratory results for an odometry only path are shown in Figure 6.36.

All fifty odometry only paths using the final calibration values are plotted in Figure 6.37. While there is a noticeable variance among these results, they qualitatively outperform the laboratory calibration results.

A quantitative analysis was performed on these odometry path errors, the results of which are seen in Table 6.35. Note that while there is a nearly a $3m$ standard deviation, there is still on average a $16m$ improvement using the SLACAM odometry calibrated values over the initial laboratory values.

The Monte-Carlo simulation and analysis shows that the SLACAM algorithm is robust to even large changes in lidar translation as well as small errors in odometry. This analysis does highlight though that a large translational error on the lidar does have an impact on the consistency of the odometry calibrated values. A SLAM only solution was generated

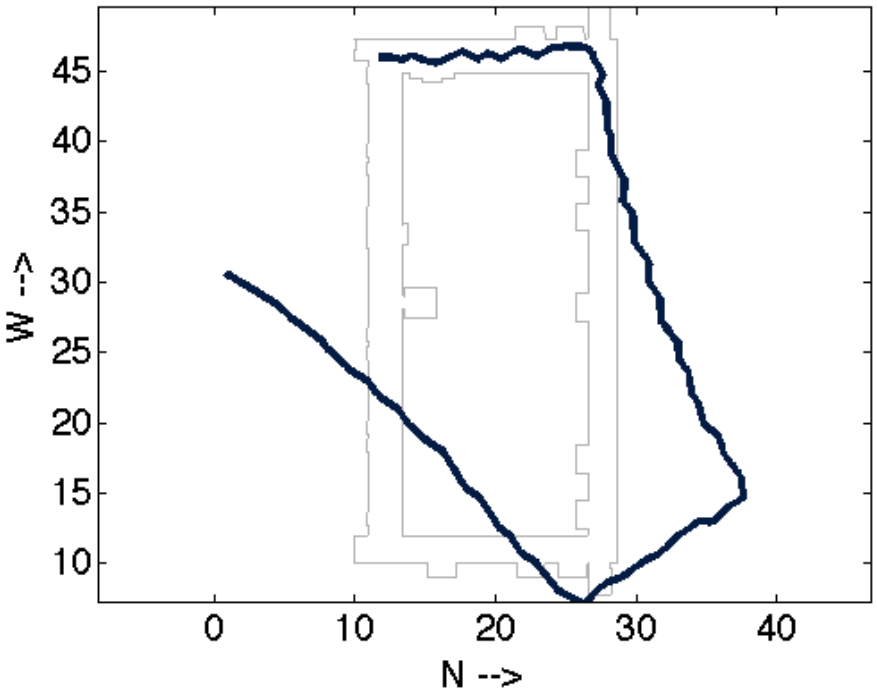

Figure 6.36: Robot Path Using Only Initial Odometry Values

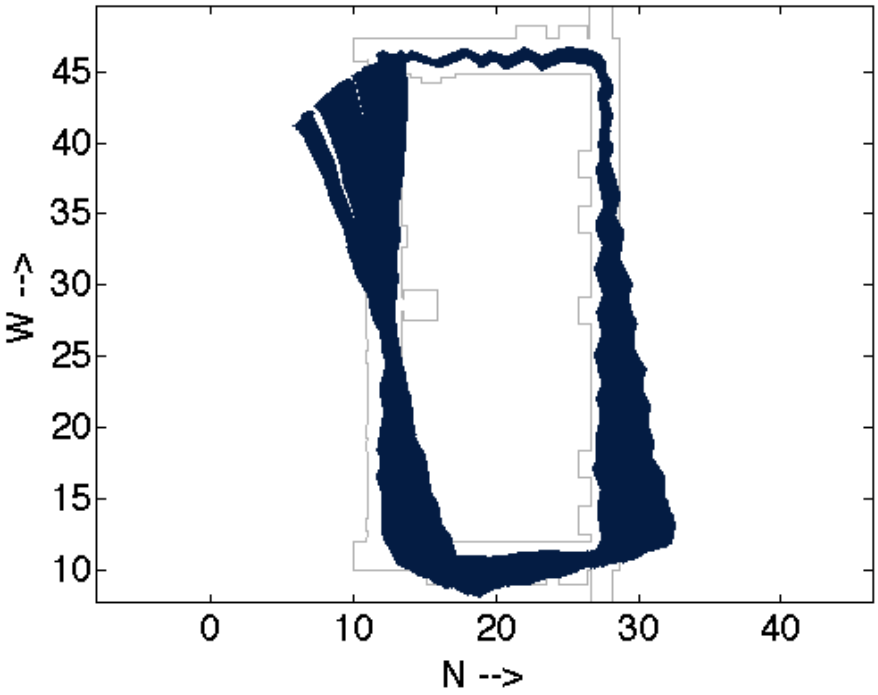

Figure 6.37: Robot Path Using Only Final Odometry Values

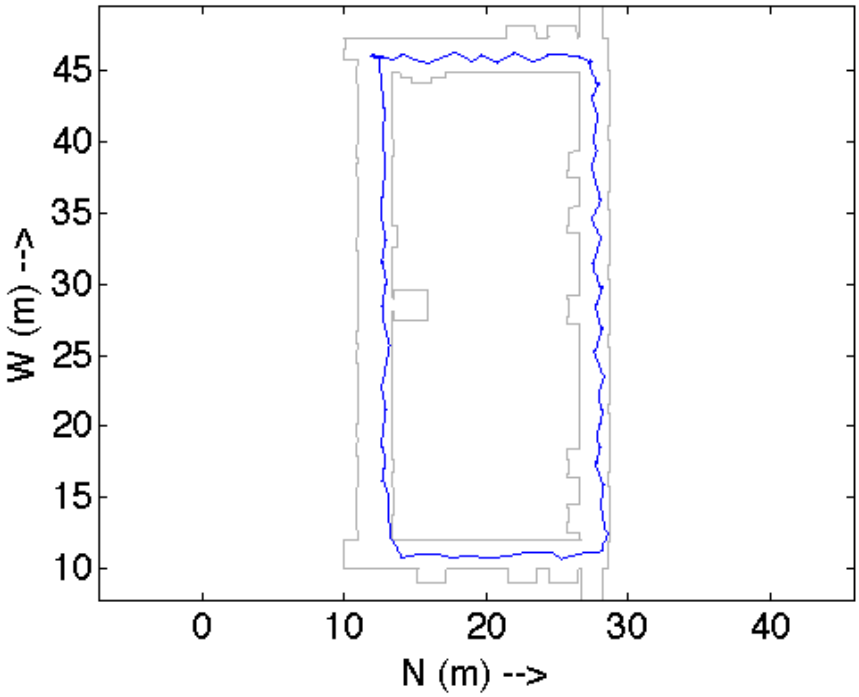

Figure 6.38: SLAM Only Solution

Table 6.36: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5453 m	-0.0094 m	$-1.6681°$

from the final calibration parameters. The path taken is seen in Figure 6.38, and the final pose errors are shown in Table 6.36.

6.4.6 Full Lidar & Odometry ICR Error

Another fifty Monte-Carlo runs focused on analyzing the ability of the SLACAM algorithm to not only accurately return to the initial position, but also on the ability to calibrate the system when all lidar and odometry ICR parameters are allowed to vary over some range is performed in this section. The wheel radii are still initialized to their laboratory calibrated values. The initial ranges for these tests are provided in Table 6.37.

The final results of these Monte-Carlo tests are provided in Table 6.38. The robot consistently returns to the initial starting location to within a length of the robot. Again the

Table 6.37: Full Lidar & ICR Error Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	$\pm\frac{0.8382m}{2}$	$\pm\frac{0.4572m}{2}$	$\pm 10°/2$	$\pm\frac{0.1m}{2}$	$\pm\frac{0.1m}{2}$	$\pm\frac{0.1m}{2}$	0m	0m

	Average Final State	Average Error	Std Error
X	N/A	0.5520m	0.0254m
Y	N/A	0.0135m	0.0148m
Θ	N/A	$-2.0957°$	$2.9431°$
X_L	0.6121m	0.0279m	0.1308m
Y_L	0.0972m	-0.0028m	0.0636m
Θ_L	$0.9328°$	$0.9328°$	$0.5484°$
X_{ICR}	-0.0741m	-0.0741m	0.0808m
Y_{ICR_L}	0.5349m	0.0516m	0.0305m
Y_{ICR_R}	-0.4563m	0.0270m	0.0302m
δ_L	0.1889m	0.0052	4.4282e-04m
δ_R	0.1874m	0.0037	5.2469e-04m

Table 6.38: Full Lidar & ICR Error Simulation Parameters - Final Errors

most noteworthy error especially as compared to Section 6.4.4 is that of a slight increase in the lidar yaw mounting error of approximately $0.33°$. Additionally, it is worth noting that there is also an increase of about $1cm$ of error in the X_{ICR} state.

Performing odometry only path solutions with the initial odometry values can ben seen in Figure 6.39. Obviously this change in the odometry parameters increases the error odometry based solution significantly.

The odometry only path using the final SLACAM odometry values can be seen in Figure 6.40. Note that these results return the robot significantly closer to the true final location.

In fact as seen in Table 6.39, there is an average path improvement of $15m$ indicating that the SLACAM solution significantly reduces the overall odometry based error.

Table 6.39: Full Lidar & ICR Simulation Parameters Odometry Values

	Error
Initial Mounting Path Error	17.79m
Final Mounting Path Error	4.14m
Odom Path Error Improvement	15.02m
Std of Odom Path Error Improvement	2.76m

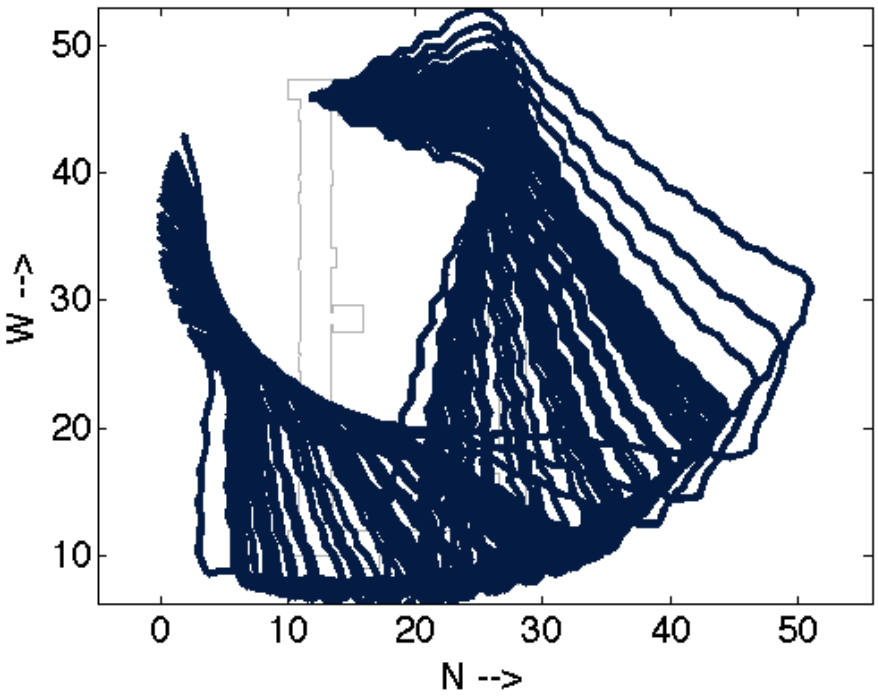

Figure 6.39: Robot Path Using Only the Initial Odometry Values

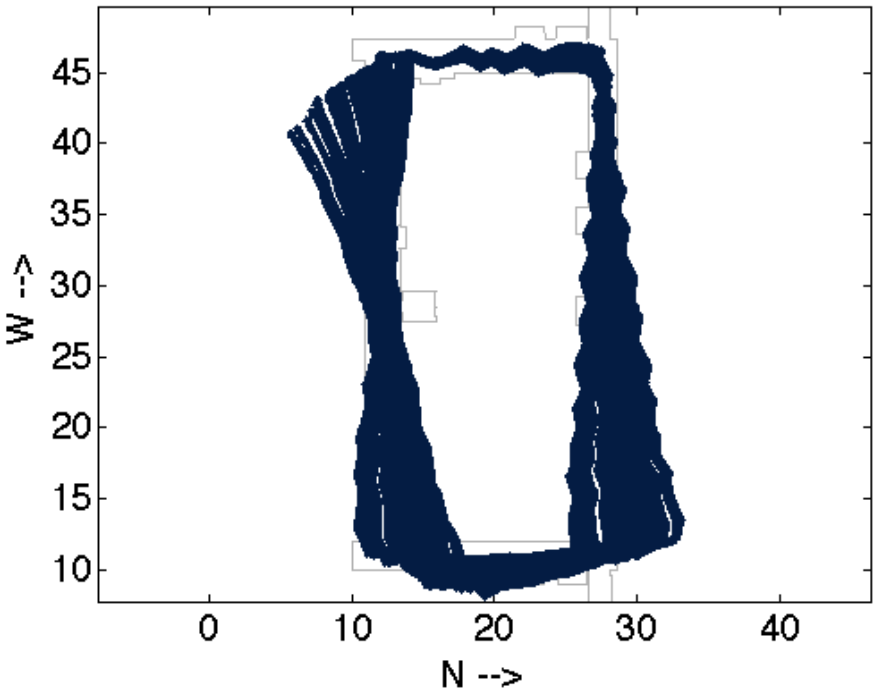

Figure 6.40: Robot Path Using Only the Final Odometry Values

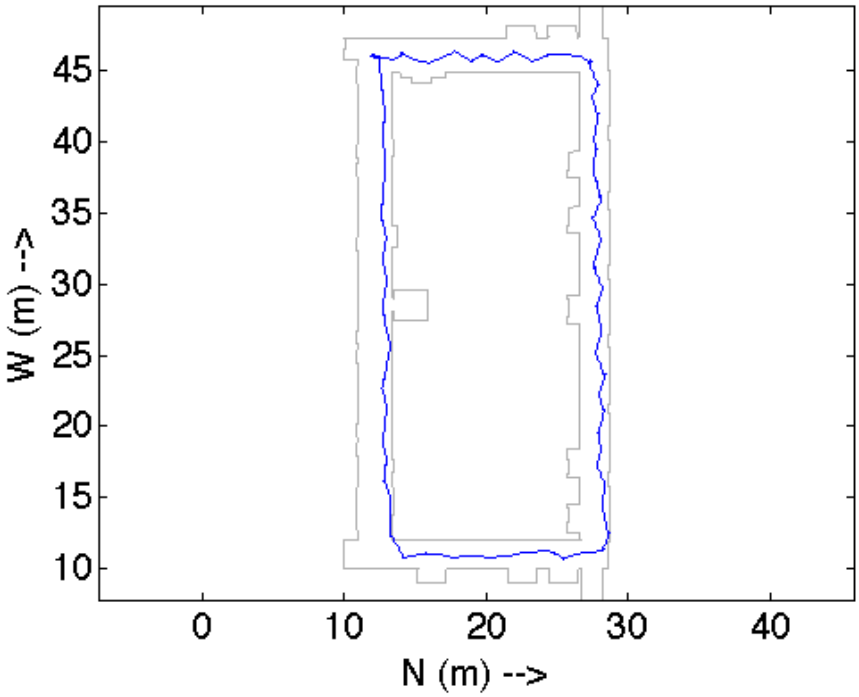

Figure 6.41: SLAM Only Solution

Table 6.40: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5452 m	-0.0077 m	$-1.6681°$

This Monte-Carlo analysis indicates that the SLACAM solution calibrate both the lidar, and odometry ICR error while successfully navigating the robot back to its initial starting location. A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.41, and the final pose errors are shown in Table 6.40.

6.4.7 Maximum Error in All Axes

Unlike the previous Monte-Carlo tests, where at least one value was always initialized to the same value, this section varies all of the parameters that SLACAM is attempting to calibrate while still localizing and mapping the environment. The full range over which parameters are allowed to vary is presented in Table 6.41.

Table 6.41: Full System Error Simulation Parameters

	X_L	Y_L	θ_L	X_{ICR}	Y_{ICR_L}	Y_{ICR_R}	δ_L	δ_R
Initial Mounting Error	$\pm\frac{0.8382m}{2}$	$\pm\frac{0.4572m}{2}$	$\pm 10°/2$	$\pm\frac{0.1m}{2}$	$\pm\frac{0.1m}{2}$	$\pm\frac{0.1m}{2}$	$\pm\frac{0.02m}{2}$	$\pm\frac{0.02m}{2}$

Table 6.42: Full System Error Simulation Parameters - Final Errors

	Average Final State	Average Error	Std Error
X	N/A	0.5517m	0.0267m
Y	N/A	0.0141m	0.0144m
Θ	N/A	$-2.0997°$	$2.9541°$
X_L	0.6121m	0.0279m	0.1308m
Y_L	0.0972m	-0.0028m	0.0636m
Θ_L	$0.9537°$	$0.9537°$	$0.5449°$
X_{ICR}	-0.0742m	-0.0742m	0.0808m
Y_{ICR_L}	0.5348m	0.0515m	0.0304m
Y_{ICR_R}	-0.4562m	0.0271m	0.0303m
δ_L	0.1889m	0.0052	5.8962e-04m
δ_R	0.1874m	0.0037	6.0141e-04m

The final results of these fifty Monte-Carlo runs is given in in Table 6.42. Note that allowing the wheel radii to vary has no appreciable change when compared to results in Section 6.4.6. Even with a full system of uncalibrated sensors, the robot is still capable of consistently returning to within a robots length of its initial starting point. Additionally all other calibration parameter errors remain on the centimeter and sub degree level.

While the final odometry values as determined via SLACAM compared to the laboratory calibrated have an error of several centimeters. From previous tests it has been shown that this is not inherently a strong indicator of SLACAM success. Qualitatively, success can be determined by comparing the initial odometry path using the values of the SLACAM result. The initial odometry paths can be seen in Figure 6.42, where it is obvious that while these odometry values varied only over a small range the path error they can cause is significant.

The odometry only path using the SLACAM calibrated odometry results are plotted in Figure 6.43. Naturally, the SLACAM results show a significant qualitative improvement over the initialized values.

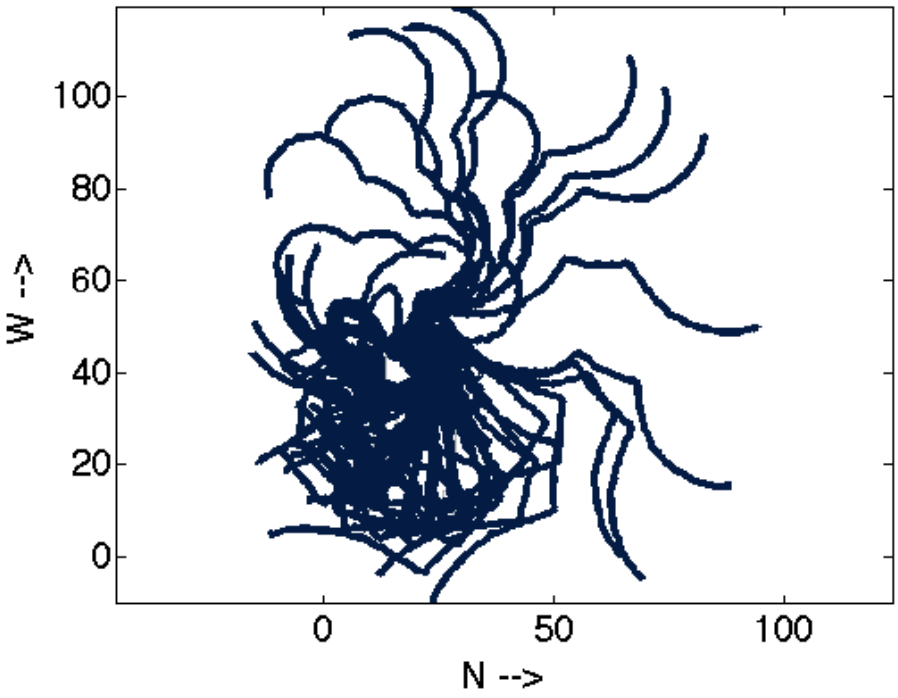

Figure 6.42: Robot Path Using Only the Initial Odometry Values

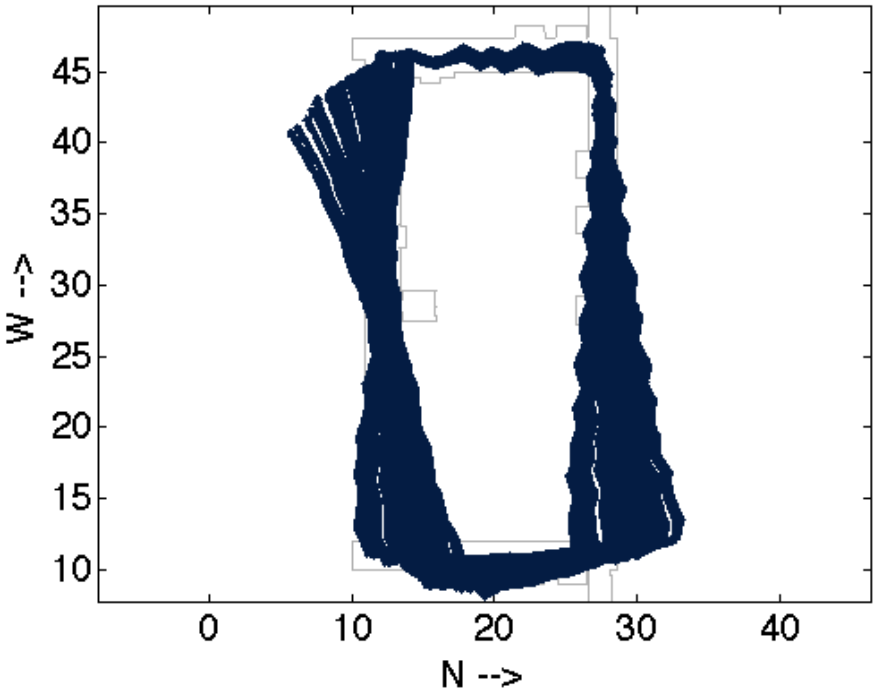

Figure 6.43: Robot Path Using Only the Final Odometry Calibration Values

Table 6.43: Full System Error Parameters Odometry Values

	Error
Initial Mounting Path Error	36.86m
Final Mounting Path Error	2.90m
Odom Path Error Improvement	33.95m
Std of Odom Path Error Improvement	23.02m

Table 6.44: SLAM Only Solution Final Position Error

X-Error	Y-Error	Θ-Error
0.5453 m	-0.0075 m	$-1.6681°$

When comparing these results quantitatively there is a path correction of nearly 34m. The standard deviation on this improvement is quite large due to the fact that the initial error varies wildly, and not because the final solution varies significantly.

From this analysis it appears that SLACAM can successfully map and localize an environment as evidence of the robot accurately returning to the initial starting location. Additionally the SLACAM algorithm can successfully simultaneously calibrate both full lidar and odometry system parameters. It is worth noting that both odometry errors and lidar translation errors impact the lidar yaw calibration performance. A SLAM only solution was generated from the final calibration parameters. The path taken is seen in Figure 6.44, and the final pose errors are shown in Table 6.44.

6.5 Conclusions

This chapter first demonstrated the ability to simulate the environment, sensors, and the SLACAM algorithm. A series of analyses was performed in simulation and using actual data that varied the various mounting values of the lidar and odometry system while also navigating through the environment and mapping it. The sensors were varied over a large range of errors and the SLACAM algorithm demonstrated that it could both accurately return to its original location despite larger initial sensor errors and successfully calibrate the sensor values. A strong agreement was shown between simulation and experimentation

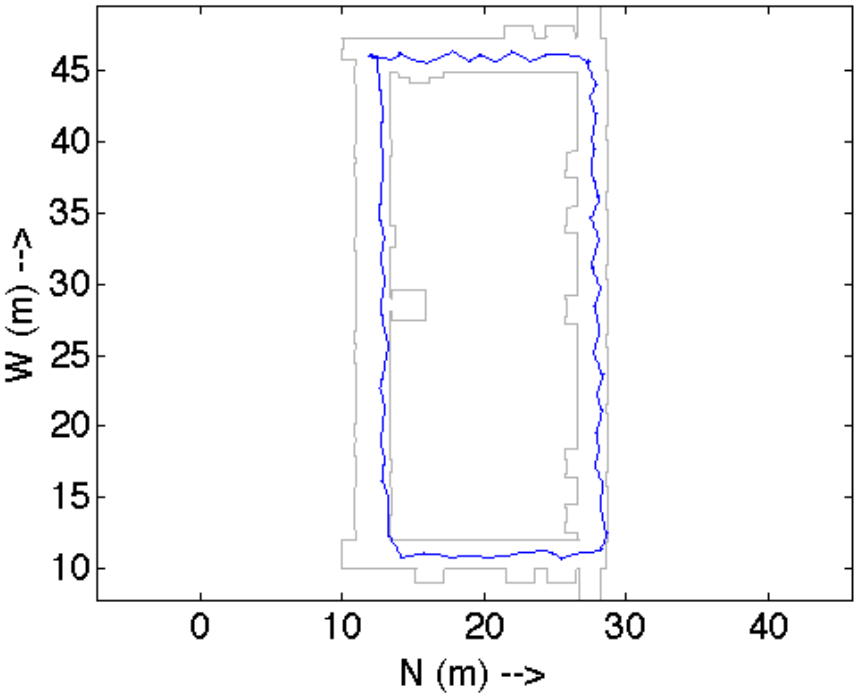

Figure 6.44: SLAM Only Solution

results. Additionally, it was shown that the SLACAM experimental results showed performance results on par with other state of the art methods that sought to only perform one of the functions of the SLACAM algorithm.

Chapter 7

Conclusions

This dissertation provided an overview of SLAM, auto-calibration, and the SLACAM problem. The prior work indicated that the calibration of two sensors often relied on the use of known maneuver, landmarks, or targets [69][87]. Additionally, these techniques often lacked a comparative truth technique and tests that involved varying initial mounting errors that covered only a narrow range [42] or provided values that where simply "optimal" but did not have a corresponding physical value to validate it against [2]. The SLACAM prior work also varied the initial mounting errors over a minimal range. Additionally, the SLACAM work almost exclusively revolved around only calibrating the parameters of a single sensor. The analysis geared towards multi-sensor SLACAM however did not implement SLACAM optimally by only looking over a narrow window of previous measurements, it was inherently not taking into account potentially beneficial information. In contrast, this dissertation provided a 3-D calibration strategy that was analyzed over a significant range of operating conditions in a realistic operating environment and independently verified through laboratory calibration. Additionally a novel implementation of SLACAM is introduced which optimally blended multiple sensors for both accurate navigation and auto-calibration.

In Chapter 3 a generic 6-DOF auto-calibration technique is introduced that relies solely on the sensors being calibrated being able to determine their relative pose. This auto-calibration technique is then formulated as a sequential estimation problem, through the use of the EKF framework. Using the sensor background information it is shown how sensors that do not inherently measure position, can still operate within the relative pose paradigm. Specifically it is shown how ICP can be used by lidar to determine relative pose by comparing sequential point clouds. The necessary formulas are also provided for

more traditional position sensors such as an IMU, INS, and wheel odometry. Note that the odometry relative pose calculations are formulated using ICRs as a skid steer robot as opposed to differential drive. This is more representative of the future ground robot used in SLACAM. Finally the observability of this 6-DOF auto-calibration technique is analyzed. From this observability analysis it shown that as long as there is more than one change in rotation in all axes, the system will be observable. Paths where no rotation occurs and paths of constant rotation are highlighted.

Validation of the previously mentioned auto-calibration technique is provided in Chapter 4. Specifically the auto-calibration of a 64 beam velodyne lidar and an Applanix INS is analyzed while it is being driven aboard an SUV in an urban environment on a path of over 25km. First the auto-calibration technique is analyzed in simulation where the vehicle path is simulated to mimc the path of the actual vehicle, the INS positions are also the same, however the lidar ICP results are simulated based off of an assumed mounting location and a percent error noise. The simulation results indicated that the algorithm could determine the correct mounting to within mm and milli degree accuracy as long as the sensor was placed on the vehicle. Simulation results where compared to the live data captured by MIT and the final auto-calibration results where compared to the laboratory calibration results also determined by MIT. Using the live data is was determined that the mounting location could still achieve cm and milli angle mounting accuracies provided that the pitch and roll where with 45° of the true value, and the yaw was within 90° of the true value. However, as the mounting angles go beyond these values degree level errors begin to occur in the angular measurements and the translation error becomes decimeter level. Additionally it is shown that if the sensor is jarred the algorithm is capable of detecting this mis-alignment and compensating for it as shown by parring simulated and real data.

Chapter 5 further extended the use of the aforementioned auto-calibration technique by combining it with SLAM to become SLACAM. The equations necessary to preform SLACAM are provided. Specifically the standard SLAM based equations are shown which expresses

how to add landmarks to the EKF state and covariance matrices. Additionally the equations are provided that demonstrated how to properly include the covariance of the various sensor that are being calibrated to the new landmark, so that as the sensor mounting confidence changes a corresponding change is seen by the landmark. The lidar based navigation equations are provided both in terms of how landmarks are extracted to aid navigation, but also how ICP is incorporated to aid in auto-calibration. The process by which the odometry system is used to both aid driving, determine when the robot is stationary to aid in IMU bias estimation, as well as odometry auto-calibration equations are provided. Once the entire SLACAM infrastructure has been developed in EKF form, the observability is analyzed drawing from Chapter 3 analysis. It is shown that while the same limitations of the auto-calibration system still apply to the SLACAM system, the system can become observable.

Finally, the SLACAM architecture is validated in Chapter 6 through both simulation and experimental data of a skid-steer robot which calibrated an IMU to both an odometry system and 2D scanning lidar operating in an indoor environment. First the an environment was constructed to represent the second floor of the Advanced Research Building where the robot would eventually collect live data. Once the environment was constructed a zig-zag path for the robot was defined from which IMU and odometry values where generated. Finally the scanning lidar output was created. This simulation data then demonstrated the ability of the SLACAM system to calibrated the lidar to within centimeter translational accuracies and sub degree accuracies for large initial translational error and small angular errors. The map generated by the SLACAM algorithm was provided which showed strong agreement to the true map used for the simulation environment. The odometry was also validated in its ability to significantly improve the initial estimate as well as show agreement with the theory that the ICRs should remain in a relatively close area. Additional simulation tests included varying the range of initial mounting errors of both the lidar and odometry systems which even at its worst, the lidar maintained cm and sub degree errors on average,

and the odometry values also averaged errors of cm level. The simulation data was then validated using an iRobot ARTV differential drive robot with a MEMS IMU and a SICK scanning lidar. Testing occurred on the second floor of the Advanced Research Building and the robot was driven in a series of zig-zag paths. The data was analyzed in post-process. The specific analysis procedure mimicked that of the simulation data as closely as possible where the initial mounting error of the sensors varied over a given range of parameters to ascertain algorithm success. It was demonstrated that when the initial values are near the true values initially, errors are of cm and sub-degree magnitudes. When initial errors are more significant where parameters are allowed to vary over the size of the robot (worst case scenario) it is shown that the errors do indeed stay bounded to cm and sub-degree magnitudes. The map of the environment generated by SLACAM is shown and compared simulated environment as qualitative evidence of success in implementing the SLAM portion of SLACAM. Finally, because the robot was driven in a path such that it ended where it began, the ability of the SLACAM algorithm to return the robot to the same initial location is analyzed as quantitative evidence. Using this metric, SLACAM demonstrated that it should return to the same starting location with an error proportional to the size of the robot.

From these results it can be concluded that SLACAM is highly accurate when the initial mounting errors are close to the true values. SLACAM can also operate robustly even when the translational errors are large, especially the odometry ICR values. This is of great benefit because this is a value that is not easily measured and can often be surface dependent. SALCAM is most sensitive to initial lidar angular mounting errors, where a 10° window around the true value is all that can be stomached before the average error is above 1°. From a purely auto-calibration perspective we can conclude that when a highly accurate ICP algorithm is used, two sensors can be calibrated to one another with a high degree of accuracy even with very large initial errors. However, it can be concluded that ideally the operator estimate the mounting angles to within 45° of the true value. The mounting

translation error does appear to be of little consequence to the final answer, as long as the operator knows that the lidar is actually mounted to within the confines of the vehicle.

Future work should include analysis of both the impact of baselines on results, as well as what impact the addition of various other sensors. Specifically for aircraft, to determine if having sensors further apart magnified any misalignment which would allow the algorithm to better detect it while simultaneously having the benefit of improving a solution under circumstances that are often the most difficult. The scenario would also benefit from GPS which could add greatly to the overall system robustness by including highly accurate velocity information to aid the change in sensor position calculations. It would also be of interest to better quantify what grade sensors are required to achieve a specific level of accuracy. While the tests analyzed both MEMs and tactical grade IMUs, the tests where not strictly a one to one comparison. Finally it would be of interest to operate the SLACAM algorithm in a scenario where the robot transition over varying surface types. Because the odometry ICP values are dependent on surface the robot is driving, it would be of interest to be able to see if it is possible to determine and classify the type of surface the robot is currently on. This information could act both as a simply classifier, as well as provide important feedback to any other safety systems the robot might be using to limit control inputs.

Bibliography

[1] Hatem Alismail, L Douglas Baker, and Brett Browning. Continuous Trajectory Estimation for 3D SLAM from Actuated Lidar *.

[2] Georgia Anousaki and KJ Kyriakopoulos. A Dead-Reckoning Scheme for Skid-Steered vehicles in Outdoor Environments. *Robotics and Automation*, (April):580–585, 2004.

[3] ME Antone and Yuli Friedman. Fully Automated Laser Range Calibration. *BMVC*, (Figure 1), 2007.

[4] P.J. Besl and H.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] Stanley Bileschi. Fully automatic calibration of LIDAR and video streams from a vehicle. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1457–1464, September 2009.

[6] Johann Borenstein and Liqiang Feng. UMBmark: A Benchmark Test for Measuring Odometry Errors in Mobile Robots. pages 113–124, December 1995.

[7] Geovany Araujo Borges and Marie-José Aldon. Line Extraction in 2D Range Images for Mobile Robotics. *Journal of Intelligent and Robotic Systems*, 40(3):267–297, July 2004.

[8] Geovany Araujo Borges and Marie-José Aldon. Line Extraction in 2D Range Images for Mobile Robotics. *Journal of Intelligent and Robotic Systems*, 40(3):267–297, July 2004.

[9] Stephen Borthwick and H. Durrant-Whyte. Dynamic localisation of autonomous guided vehicles. In *Proceedings of 1994 IEEE International Conference on MFI '94. Multisensor Fusion and Integration for Intelligent Systems*, pages 92–97. IEEE, 1994.

[10] Briask. Applanix Media, 2012.

[11] D. Britt, J., Bevly. Sensor Auto-Calibration on Dynamic Platforms in 3D. *Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013)*, pages 2195–2203, 2013.

[12] Jonathan Brookshire and Seth Teller. Automatic calibration of multiple coplanar sensors. *Robotics: Science and Systems VII*, 2012.

[13] Jonathan Brookshire and Seth Teller. Extrinsic Calibration from Per-Sensor Egomotion. *Proceedings of Robotics Science and Systems*, 2012.

[14] D. Caltabiano. Localization and self-calibration of a robot for volcano exploration. *Robotics and Automation*, pages 586–591 Vol.1, 2004.

[15] J.a. Castellanos, J. Neira, and J.D. Tardos. Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 17(6):908–914, 2001.

[16] Andrea Censi, Luca Marchionni, and Giuseppe Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. *2008 IEEE International Conference on Robotics and Automation*, pages 2098–2103, May 2008.

[17] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, (April):2724–2729, 1991.

[18] Harald Cramér. *Mathematical Methods of Statistics*. 1946.

[19] Travis Deyle. Hizook - MIT's DARPA Urban Grand Challenge, 2008.

[20] Albert Diosi and Lindsay Kleeman. *Uncertainty of line segments extracted from static sick pls laser scans*. 2003.

[21] H Durrant-Whyte and T Bailey. Simultaneous localization and mapping: part I. *Robotics & Automation Magazine*, 2006.

[22] H.F. Durrant-Whyte. Uncertain geometry in robotics. *Robotics and Automation, IEEE Journal of*, 4(1):23–31, 1988.

[23] Tobias Einsele. Real-time self-localization in unknown indoor environment using a panorama laser range finder. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97*, volume 2, pages 697–702. IEEE, 1997.

[24] A Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 1989.

[25] Adam Feldman, Tucker Balch, and Rick Cavallaro. The Multi-ICP Tracker : An Online Algorithm for Tracking Multiple Interacting Targets.

[26] W Flenniken. *MODELING INERTIAL MEASUREMENT UNITS AND ANALYZING THE EFFECT OF THEIR ERRORS IN NAVIGATION APPLICATIONS*. PhD thesis, Auburn University, 2005.

[27] EM Foxlin. Generalized architecture for simultaneous localization, auto-calibration, and map-building. *Intelligent Robots and Systems, 2002. IEEE/RSJ*, (October):527–533, 2002.

[28] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, April 2005.

[29] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds. *Information Fusion*, (3), July 2011.

[30] Dr Rainer Hessmer. Hessmer Blog, 2011.

[31] AS Huang, M. Antone, and E. Olson. A high-rate, heterogeneous data set from the darpa urban challenge. *Journal of Robotics*, 29(13):1595–1601, November 2010.

[32] T. Kluge J. Scholz, V. Willhoeft, Dr. R. Schulz. ALASCA ÂⓇ User Manual. 2006.

[33] Isprs Journal and Remote Sensing. Least squares 3D surface and curve matching Armin Gruen , Devrim Akca*. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):151–174, 2005.

[34] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, pages 360–368, December 2009.

[35] W.S. Kim, A.I. Ansar, and R.D. Steele. Rover mast calibration, exact camera pointing, and camera handoff for visual target tracking. *Advanced Robotics, 2005. ICAR ...*, (2):384–391, 2005.

[36] L Kleeman. Accurate odometry and error modelling for a mobile robot. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 2783–2788. IEEE, 1997.

[37] Rainer Kummerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 26, pages 3716–3721. IEEE, September 2011.

[38] Rainer Kummerle, Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Simultaneous parameter calibration, localization, and mapping for robust service robotics. *Advanced Robotics and its Social Impacts*, pages 76–79, October 2011.

[39] Se-jin Lee, Jae-bok Song, A Footprint Association, and F P A Model. A new sonar salient feature structure for EKF-based SLAM. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5966–5971. IEEE, October 2010.

[40] YH Lee, Changhyun Jun, and HA Choi. The analysis of effect of observation models and data associations on the consistency of EKF SLAM. *Ubiquitous Robots and ...*, pages 359–362, 2011.

[41] John Leonard, David Barrett, Jonathan How, and Seth Teller. Team MIT Urban Challenge Technical Report. Technical report, 2007.

[42] Jesse Levinson and Sebastian Thrun. Unsupervised Calibration for Multi-beam Lasers. *Experimental Robotics*, 2014.

[43] M Li, W Li, Jian Wang, Qingquan Li, and Andreas Nüchter. Dynamic VeloSLAM – Preliminary Report on 3D Mapping of Dynamic Environments. *IEEE Intelligent Vehicles Symposium*, pages 1–6, 2012.

[44] Kok-Lim Low. Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. *Technical Report TR04-004, Department of Computer Science, University of North Carolina at Chapel Hill*, (February), 2004.

[45] Will Maddern, Alastair Harrison, and Paul Newman. Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs. *2012 IEEE International Conference on Robotics and Automation*, pages 3096–3102, May 2012.

[46] Anthony Mandow and JL Martinez. Experimental kinematics for wheeled skid-steer mobile robots. *Intelligent Robots*, 2007.

[47] A. Martinelli, D. Scaramuzza, and R. Siegwart. Automatic self-calibration of a vision system during robot motion. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, (May):43–48, 2006.

[48] A Martinelli and Nicola Tomatis. Simultaneous localization and odometry calibration for mobile robot. *Intelligent Robots and Systems*, (October), 2003.

[49] A Martinelli, Nicola Tomatis, and R Siegwart. Simultaneous localization and odometry self calibration for mobile robot. *Autonomous Robots*, (October), 2007.

[50] Agostino Martinelli and Roland Siegwart. Observability Properties and Optimal Trajectories for On-line Self-Calibration. *Proceedings of the IEEE Conference on . . .*, pages 3065–3070, 2006.

[51] J. L. Martinez. Approximating Kinematics for Tracked Mobile Robots. *The International Journal of Robotics Research*, 24(10):867–878, October 2005.

[52] JL Martinez and A Mandow. Kinematic modelling of tracked vehicles by experimental identification. *Robots and Systems*, 2004.

[53] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis. 3D LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The International Journal of Robotics Research*, 31(4):452–467, April 2012.

[54] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem.

[55] Peter Morton. Research - Australian Centre For Field Robotics. 2010.

[56] Peter Morton. Peter Morton Research, 2015.

[57] RL Moses, R Patterson, and Wendy Garber. Self localization of acoustic sensor networks. *Military Sensing Symposia (MSS) Specialty Group on Battlefield Acoustic and Seismic Sensing, Magnetic and Electric Field Sensors*, 2002.

[58] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[59] V. Nguyen and A. Martinelli. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. *Intelligent Robots and ...*, pages 1929–1934, 2005.

[60] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots*, 23(2):97–111, June 2007.

[61] P. Núñez, R. Vázquez-Martín, J.C. del Toro, A. Bandera, and F. Sandoval. Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. *Robotics and Autonomous Systems*, 56(3):247–264, March 2008.

[62] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2262–2269.

[63] Albert Palomer, Pere Ridao, David Ribas, Angelos Mallios, Nuno Gracias, and Guillem Vallicrosa. Bathymetry-based SLAM with difference of normals point-cloud subsampling and probabilistic ICP registration. *2013 MTS/IEEE OCEANS - Bergen*, pages 1–8, June 2013.

[64] Gaurav Pandey, JR McBride, Silvio Savarese, and R Eustice. Automatic Targetless Extrinsic Calibration of a 3D Lidar and Camera by Maximizing Mutual Information. *AAAI National Conference on Artifical Intelligence*, (2005), 2012.

[65] Theodosios Pavlidis and S.L. Horowitz. Segmentation of Plane Curves. *IEEE Transactions on Computers*, C-23(8):860–870, August 1974.

[66] L.M. Paz, J.D. Tardos, and J. Neira. Divide and Conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107–1120, October 2008.

[67] S.T. Pfister. Weighted range sensor matching algorithms for mobile robot displacement estimation. *Robotics and Automation*, 2(May):1667–1674, 2002.

[68] S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 1:1304–1311, 2003.

[69] A Pothou, C Toth, S Karamitos, and A Georgopoulos. On using QA/QC techniques for LiDAR/IMU boresight misalignment. *International Symposium on Mobile Mapping Technology*, 2007.

[70] Li Qin, Wang; Chris, Rizos; Yong, Li; Shiyi. Application of a sigma-point Kalman filter for alignment of MEMS-IMU. *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 44–52, 2008.

[71] Radhakrishna C. Rao. Information and the Accuracy Attainable in the Estimation of Statistical Parameters. *Bulletin of Cal. Math. Soc.*, 37(3):81–91, 1945.

[72] D Rodriguez-Losada and J Minguez. Improved Data Association for ICP-based Scan Matching in Noisy and Dynamic Environments. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3161–3166, April 2007.

[73] RM Rogers. Weapon IMU Transfer Alignment Using Aircraft Position from Acutal Flight Tests. *Position Location and Navigation Symposium*, 1996.

[74] RM Rogers. Low Dynamic IMU Alignment. *Position Location and Navigation Symposium*, pages 272–279, 1998.

[75] Hyun Chul Roh, Chang Hun Sung, Min Tae Kang, and Myung Jin Chung. Fast SLAM using polar scan matching and particle weight based occupancy grid map for mobile robot. *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, (2):756–757, November 2011.

[76] N. Roy and S. Thrun. Online self-calibration for mobile robots. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 3:2292–2297.

[77] Alexander Rudolph. Quantification and Estimation of Differential Odometry Errors in Mobile Robotics with Redundant Sensor Information. *The International Journal of Robotics Research*, 22(2):117–128, February 2003.

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

[79] A Siadat and A Kaske. AN OPTIMIZED SEGMENTATION METHOD FOR A 2D LASER-SCANNER APPLIED TO MOBILE ROBOT. *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, (1):2–7, 1997.

[80] David A Simon, Michael Erdmann, and Eric Grimson. Fast and Accurate Shape-Based Registration. 1996.

[81] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. *Autonomous robot vehicles*, 1990.

[82] Randall C Smith, Medo Park, and Peter Cheeseman. On the Representation and Estimation of Spatial Uncertainty.

[83] RF Stengel. *Optimal control and estimation*. 1986.

[84] C. Taylor, a. Rahimi, J. Bachrach, H. Shrobe, and a. Grue. Simultaneous localization, calibration, and tracking in an ad hoc sensor network. *2006 5th International Conference on Information Processing in Sensor Networks*, pages 27–33, 2006.

[85] J Taylor. The Cramer-Rao Estimation Error Lower Bound Computation for Deterministic Nonlinear Systems. *Automatic Control, IEEE Transactions on*, (2):343–344, 1979.

[86] S. Thrun. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429, May 2006.

[87] James Underwood, Andrew Hill, and Steve Scheding. Calibration of range sensor pose on mobile platforms. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3866–3871, October 2007.

[88] R. Wagner, O. Birbach, and U. Frese. Rapid development of manifold-based graph optimization systems for multi-sensor calibration and SLAM. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3305–3312, September 2011.

[89] SICK AG Waldkirch. LMS200/211/221/291 Laser Measurement Systems Technical Description.

[90] Zhan Wang and Gamini Dissanayake. Observability analysis of SLAM using fisher information matrix. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, number December, pages 1242–1247. IEEE, December 2008.

[91] J.W. Weingarten, G. Gruener, and R. Siegwart. Probabilistic plane fitting in 3D and an application to robotic mapping. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 927–932 Vol.1, 2004.

[92] D Wittman. Fisher Matrix for Beginners.

[93] J. Y. Wong and C. F. Chiang. A general theory for skid steering of tracked vehicles on firm ground. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 215(3):343–355, January 2001.

[94] Jingang Yi, Hongpeng Wang, and Junjie Zhang. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *Robotics, IEEE ...*, 25(5):1087–1097, 2009.

[95] Jingang Yi, Junjie Zhang, Dezhen Song, and Suhada Jayasuriya. IMU-based localization and slip estimation for skid-steered mobile robots. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2845–2850, October 2007.

[96] Wende Zhang. LIDAR-Based Road and Road-Edge Detection. *Integration The Vlsi Journal*, pages 845–848, 2010.

[97] Zhenqi Zhu, Qing Tang, Jinsong Li, and Zhongxue Gan. Calibration of laser displacement sensor used by industrial robots. *Optical Engineering*, 43(1):12, 2004.

[98] Zhu Zhu and Jilin Liu. Unsupervised Extrinsic Parameters Calibration for Multi-beam LIDARs. *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, (Iccsee):1110–1113, 2013.

Appendices

Appendix A

Observability of Single Delta Pose Measurement

This appendix will demonstrated that a single observation in the change in pose of both the reference sensor and the sensor to be calibrates lacks sufficient information to be observable. The pertinent measurement equations are reproduced below from Equations(3.5,3.11,3.9).

$$\begin{aligned} \hat{\delta}^{t,s}_{t-1,s} &= R^s_v \left(\delta^{t,v}_{t-1,v} + R^{t,v}_{t-1,v} S^v_\circ - S^v_\circ \right) \\ \hat{R}^{t,s}_{t-1,s} &= R^s_v R^{t,v}_{t-1,v} R^v_s \end{aligned} \tag{A.1}$$

$$G = \begin{bmatrix} \begin{bmatrix} \hat{\delta}^{t,s}_{t-1,s} \end{bmatrix} \\ \arctan 2 \left(\frac{\hat{R}^{t,s}_{t-1,s}(2,1)}{\hat{R}^{t,s}_{t-1,s}(1,1)} \right) \\ \arctan 2 \left(\frac{-\hat{R}^{t,s}_{t-1,s}(3,1)}{\sqrt{\left(\hat{R}^{t,s}_{t-1,s}(3,2)\right)^2 + \left(\hat{R}^{t,s}_{t-1,s}(3,3)\right)^2}} \right) \\ \arctan 2 \left(\frac{\hat{R}^{t,s}_{t-1,s}(3,2)}{\hat{R}^{t,s}_{t-1,s}(3,3)} \right) \end{bmatrix} \tag{A.2}$$

$$H_k = \frac{\partial}{\partial \mathbf{X}_k} G \left(\delta^{t,v}_{t-1,v},\ R^{t,v}_{t-1,v},\ \mathbf{X_k} \right) \tag{A.3}$$

Recall that for a linear system to be observable H_k must have full column rank. Additionally recall that for full column rank to exist, H_k must also have a non-zero determinant. If H_k does have a determinant which is zero, this implies a linear dependance amongst equations. The determinant of H_k is seen below with the aid of defining the following constants to aid in illustrating which terms will cancel out.

$$AAA = \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.4}$$

$$AAB = \cos(\theta) * \cos(\phi) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^3 \tag{A.5}$$

$$AAC = \cos(\phi) * \cos(\psi) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \tag{A.6}$$

$$AAD = \cos(\theta) * \cos(\phi)^2 * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v) \tag{A.7}$$

$$AAE = \cos(\theta) * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \tag{A.8}$$

$$AAF = \cos(\theta)^2 * \cos(\phi)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.9}$$

$$AAG = \cos(\theta)^2 * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.10}$$

$$AAH = \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \tag{A.11}$$

$$AAI = \cos(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.12}$$

$$AAJ = \cos(\phi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.13}$$

$$AAK = \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.14}$$

$$AAL = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^3 * \sin(\phi) * \sin(\phi_v) \tag{A.15}$$

$$AAM = \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.16}$$

$$AAN = \cos(\theta)^3 * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) \tag{A.17}$$

$$AAO = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.18}$$

$$AAP = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \tag{A.19}$$

$$AAQ = \cos(\theta) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \tag{A.20}$$

$$AAR = \cos(\theta) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \tag{A.21}$$

$$AAS = \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \tag{A.22}$$

$$AAT = \cos(\theta) * \cos(\psi) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.23}$$

$$AAU = \cos(\theta) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.24}$$

$$AAV = \cos(\theta_v) * \cos(\phi) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.25}$$

$$AAW = \cos(\theta)^3 * \cos(\phi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.26}$$

$$AAX = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\psi) * \sin(\psi_v) \tag{A.27}$$

$$AAY = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^3 \tag{A.28}$$

$$AAZ = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\psi) * \sin(\psi_v) \tag{A.29}$$

$$ABA = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^2 \tag{A.30}$$

$$ABB = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\psi_v) \tag{A.31}$$

$$ABC = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.32}$$

$$ABD = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.33}$$

$$ABE = \cos(\theta) * \cos(\theta_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \tag{A.34}$$

$$ABF = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.35}$$

$$ABG = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.36}$$

$$ABH = 2 * \cos(\theta) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^2 \tag{A.37}$$

$$ABI = \cos(\phi) * \cos(\phi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.38}$$

$$ABJ = \cos(\theta_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.39}$$

$$ABK = \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.40}$$

$$ABL = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \tag{A.41}$$

$$ABM = \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \tag{A.42}$$

$$ABN = \cos(\theta) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.43}$$

$$ABO = 2 * \cos(\theta) * \cos(\phi) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.44}$$

$$ABP = \cos(\phi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.45}$$

$$ABQ = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) \tag{A.46}$$

$$ABR = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) \tag{A.47}$$

$$ABS = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) \tag{A.48}$$

$$ABT = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\phi) * \sin(\phi_v) \tag{A.49}$$

$$ABU = \cos(\theta) * \cos(\theta_v) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \tag{A.50}$$

$$ABV = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \tag{A.51}$$

$$ABW = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \tag{A.52}$$

$$ABX = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \tag{A.53}$$

$$ABY = \cos(\theta) * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \tag{A.54}$$

$$ABZ = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \tag{A.55}$$

$$ACA = \cos(\theta) * \cos(\phi) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) \tag{A.56}$$

$$ACB = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.57}$$

$$ACC = 2 * \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \tag{A.58}$$

$$ACD = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \tag{A.59}$$

$$ACE = 3 * \cos(\theta) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.60)}$$

$$ACF = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.61)}$$

$$ACG = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.62)}$$

$$ACH = \cos(\theta) * \cos(\theta_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.63)}$$

$$ACI = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.64)}$$

$$ACJ = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.65)}$$

$$ACK = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.66)}$$

$$ACL = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.67)}$$

$$ACM = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.68)}$$

$$ACN = \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.69)}$$

$$ACO = 3 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.70)}$$

$$ACP = 2 * \cos(\theta)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.71)}$$

$$ACQ = 2 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.72)}$$

$$ACR = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.73)}$$

$$ACS = \cos(\theta) * \cos(\theta_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.74)}$$

$$ACT = \cos(\theta) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.75)}$$

$$ACU = \cos(\phi) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.76)}$$

$$ACV = \cos(\theta) * \cos(\phi) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.77)}$$

$$ACW = \cos(\phi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.78)}$$

$$ACX = 2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.79)}$$

$$ACY = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.80)}$$

$$ACZ = \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.81)}$$

$$ADA = \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.82)}$$

$$ADB = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) \quad \text{(A.83)}$$

$$ADC = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.84)}$$

$$ADD = \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.85)}$$

$$ADE = \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.86)}$$

$$ADF = \cos(\theta) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 \quad \text{(A.87)}$$

$$ADG = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\psi_v)^2 \quad \text{(A.88)}$$

$$ADH = \cos(\theta) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \quad \text{(A.89)}$$

$$ADI = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.90)}$$

$$ADJ = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^3 * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.91)}$$

$$ADK = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.92)}$$

$$ADL = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.93)}$$

$$ADM = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.94)}$$

$$ADN = \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.95)}$$

$$ADO = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.96)}$$

$$ADP = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.97)}$$

$$ADQ = \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.98)}$$

$$ADR = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.99)}$$

$$ADS = \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.100)}$$

$$ADT = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.101)}$$

$$ADU = \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.102)}$$

$$ADV = \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.103)}$$

$$ADW = \cos(\theta) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 \quad \text{(A.104)}$$

$$ADX = \cos(\theta) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi_v)^2 \quad \text{(A.105)}$$

$$ADY = \cos(\theta) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi_v)^2 \quad \text{(A.106)}$$

$$ADZ = \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.107)}$$

$$AEA = \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi)^3 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.108)}$$

$$AEB = \cos(\theta) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.109)}$$

$$AEC = \cos(\theta) * \cos(\phi)^2 * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.110)}$$

$$AED = \cos(\theta)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.111)}$$

$$AEE = \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.112)}$$

$$AEF = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.113)}$$

$$AEG = \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.114)}$$

$$AEH = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.115)}$$

$$AEI = \cos(\theta)^3 * \cos(\phi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.116)}$$

$$AEJ = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.117)}$$

$$AEK = \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.118)}$$

$$AEL = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.119)}$$

$$AEM = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.120)}$$

$$AEN = \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.121)}$$

$$AEO = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.122)}$$

$$AEP = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.123)}$$

$$AEQ = \cos(\theta_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.124)}$$

$$AER = \cos(\theta) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2 \quad \text{(A.125)}$$

$$AES = \cos(\theta) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.126)}$$

$$AET = \cos(\theta) * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.127)}$$

$$AEU = \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.128)}$$

$$AEV = \cos(\theta_v) * \cos(\phi) * \sin(\theta)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.129)}$$

$$AEW = \cos(\theta)^3 * \cos(\phi) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.130)}$$

$$AEX = \cos(\theta_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.131)}$$

$$AEY = \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.132)}$$

$$AEZ = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) \quad \text{(A.133)}$$

$$AFA = \cos(\theta) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \quad \text{(A.134)}$$

$$AFB = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^3 \quad \text{(A.135)}$$

$$AFC = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.136)}$$

$$AFD = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.137)}$$

$$AFE = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.138)}$$

$$AFF = \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^3 \quad \text{(A.139)}$$

$$AFG = \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.140)}$$

$$AFH = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.141)}$$

$$AFI = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.142)}$$

$$AFJ = \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.143)}$$

$$AFK = \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.144)}$$

$$AFL = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.145)}$$

$$AFM = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \quad \text{(A.146)}$$

$$AFN = \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.147)}$$

$$AFO = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.148)}$$

$$AFP = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.149)}$$

$$AFQ = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^3 \quad \text{(A.150)}$$

$$AFR = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.151)}$$

$$AFS = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.152)}$$

$$AFT = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^2 \quad \text{(A.153)}$$

$$AFU = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^2 \quad \text{(A.154)}$$

$$AFV = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v)^3 * \sin(\psi_v) \quad \text{(A.155)}$$

$$AFW = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.156)}$$

$$AFX = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.157)}$$

$$AFY = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.158)}$$

$$AFZ = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.159)}$$

$$AGA = \cos(\theta) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.160)}$$

$$AGB = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.161)}$$

$$AGC = 3 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.162)}$$

$$AGD = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^3 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.163)}$$

$$AGE = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.164)}$$

$$AGF = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.165)}$$

$$AGG = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.166)}$$

$$AGH = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.167)}$$

$$AGI = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.168)}$$

$$AGJ = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^3 * \sin(\psi_v)^2 \quad \text{(A.169)}$$

$$AGK = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.170)}$$

$$AGL = \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.171)}$$

$$AGM = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^2 \quad \text{(A.172)}$$

$$AGN = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.173)}$$

$$AGO = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^2 \quad \text{(A.174)}$$

$$AGP = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.175)}$$

$$AGQ = \cos(\theta) * \cos(\theta_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v)^3 * \sin(\psi_v) \quad \text{(A.176)}$$

$$AGR = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.177)}$$

$$AGS = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.178)}$$

$$AGT = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.179)}$$

$$AGU = \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.180)}$$

$$AGV = \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \quad \text{(A.181)}$$

$$AGW = \cos(\theta)^3 * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.182)}$$

$$AGX = \cos(\theta_v)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.183)}$$

$$AGY = \cos(\theta) * \cos(\theta_v)^3 * \cos(\phi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.184)}$$

$$AGZ = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.185)}$$

$$AHA = 3 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.186)}$$

$$AHB = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.187)}$$

$$AHC = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.188)}$$

$$AHD = \cos(\theta_v) * \cos(\phi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.189)}$$

$$AHE = \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.190)}$$

$$AHF = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.191)}$$

$$AHG = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.192)}$$

$$AHH = \cos(\theta) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.193)}$$

$$AHI = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.194)}$$

$$AHJ = \cos(\theta_v) * \cos(\phi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \quad \text{(A.195)}$$

$$AHK = \cos(\theta) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.196)}$$

$$AHL = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) \quad \text{(A.197)}$$

$$AHM = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) \quad \text{(A.198)}$$

$$AHN = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.199)}$$

$$AHO = \cos(\theta)^3 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.200)}$$

$$AHP = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \quad \text{(A.201)}$$

$$AHQ = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v) * \sin(\psi_v)^2 \quad \text{(A.202)}$$

$$AHR = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^3 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\psi_v)^2 \quad \text{(A.203)}$$

$$AHS = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 \quad \text{(A.204)}$$

$$AHT = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.205)}$$

$$AHU = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.206)}$$

$$AHV = \cos(\theta)^2 * \cos(\theta_v)^3 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.207)}$$

$$AHW = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.208)}$$

$$AHX = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.209)}$$

$$AHY = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.210)}$$

$$AHZ = \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.211)}$$

$$AIA = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.212)}$$

$$AIB = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.213)}$$

$$AIC = \cos(\theta)^3 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.214)}$$

$$AID = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.215)}$$

$$AIE = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.216)}$$

$$AIF = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.217)}$$

$$AIG = \cos(\theta) * \cos(\theta_v)^3 * \cos(\phi)^2 * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.218)}$$

$$AIH = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.219)}$$

$$AII = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.220)}$$

$$AIJ = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.221)}$$

$$AIK = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.222)}$$

$$AIL = \cos(\theta_v)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.223)}$$

$$AIM = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.224)}$$

$$AIN = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^3 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi_v)^2 \quad \text{(A.225)}$$

$$AIO = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v)^3 * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi_v)^2 \quad \text{(A.226)}$$

$$AIP = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.227)}$$

$$AIQ = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.228)}$$

$$AIR = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.229)}$$

$$AIS = \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.230)}$$

$$AIT = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.231)}$$

$$AIU = \cos(\theta)^2 * \cos(\theta_v)^3 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.232)}$$

$$AIV = \cos(\theta_v)^3 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.233)}$$

$$AIW = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.234)}$$

$$AIX = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.235)}$$

$$AIY = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.236)}$$

$$AIZ = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 \quad \text{(A.237)}$$

$$AJA = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.238)}$$

$$AJB = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.239)}$$

$$AJC = \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.240)}$$

$$AJD = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.241)}$$

$$AJE = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.242)}$$

$$AJF = 3 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi_v) \quad \text{(A.243)}$$

$$AJG = \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.244)}$$

$$AJH = \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.245)}$$

$$AJI = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.246)}$$

$$AJJ = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.247)}$$

$$AJK = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.248)}$$

$$AJL = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.249)}$$

$$AJM = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.250)}$$

$$AJN = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.251)}$$

$$AJO = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.252)}$$

$$AJP = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.253)}$$

$$AJQ = \cos(\theta) * \cos(\theta_v)^3 * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.254)}$$

$$AJR = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.255)}$$

$$AJS = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.256)}$$

$$AJT = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.257)}$$

$$AJU = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \quad \text{(A.258)}$$

$$AJV = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.259)}$$

$$AJW = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.260)}$$

$$AJX = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.261)}$$

$$AJY = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.262)}$$

$$AJZ = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.263)}$$

$$AKA = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.264)}$$

$$AKB = \cos(\theta) * \cos(\phi) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.265)}$$

$$AKC = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.266)}$$

$$AKD = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.267)}$$

$$AKE = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.268)}$$

$$AKF = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.269)}$$

$$AKG = 2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.270)}$$

$$AKH = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.271)}$$

$$AKI = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.272)}$$

$$AKJ = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.273)}$$

$$AKK = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.274)}$$

$$AKL = \cos(\theta_v) * \cos(\phi_v)^3 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.275)}$$

$$AKM = \cos(\theta_v)^3 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.276)}$$

$$AKN = \cos(\theta) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.277)}$$

$$AKO = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.278)}$$

$$AKP = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.279)}$$

$$AKQ = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi_v)^2 \quad \text{(A.280)}$$

$$AKR = \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \quad \text{(A.281)}$$

$$AKS = \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.282)}$$

$$AKT = \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.283)}$$

$$AKU = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.284)}$$

$$AKV = \cos(\phi) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.285)}$$

$$AKW = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.286)}$$

$$AKX = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.287)}$$

$$AKY = 2 * \cos(\theta) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.288)}$$

$$AKZ = \cos(\theta_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.289)}$$

$$ALA = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.290)}$$

$$ALB = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.291)}$$

$$ALC = \cos(\theta) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.292)}$$

$$ALD = \cos(\theta) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.293)}$$

$$ALE = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.294)}$$

$$ALF = \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.295)}$$

$$ALG = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.296)}$$

$$ALH = \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.297)}$$

$$ALI = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.298)}$$

$$ALJ = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.299)}$$

$$ALK = \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.300)}$$

$$ALL = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.301)}$$

$$ALM = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.302)}$$

$$ALN = \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \quad \text{(A.303)}$$

$$ALO = \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.304)}$$

$$ALP = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.305)}$$

$$ALQ = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.306)}$$

$$ALR = \cos(\theta) * \cos(\theta_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.307)}$$

$$ALS = 2 * \cos(\theta) * \cos(\phi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.308)}$$

$$ALT = \cos(\theta) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.309)}$$

$$ALU = \cos(\theta_v) * \cos(\phi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.310)}$$

$$ALV = \cos(\phi) * \cos(\phi_v) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \quad \text{(A.311)}$$

$$ALW = \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.312)}$$

$$ALX = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.313)}$$

$$ALY = 2 * \cos(\theta) * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.314)}$$

$$ALZ = \cos(\theta_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.315)}$$

$$AMA = \cos(\theta_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.316)}$$

$$AMB = \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.317)}$$

$$AMC = \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.318)}$$

$$AMD = \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.319)}$$

$$AME = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.320)}$$

$$AMF = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.321)}$$

$$AMG = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \quad \text{(A.322)}$$

$$AMH = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.323)}$$

$$AMI = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.324)}$$

$$AMJ = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.325)}$$

$$AMK = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.326)}$$

$$AML = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.327)}$$

$$AMM = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.328)}$$

$$AMN = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \quad \text{(A.329)}$$

$$AMO = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.330)}$$

$$AMP = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.331)}$$

$$AMQ = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.332)}$$

$$AMR = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.333)}$$

$$AMS = \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \quad \text{(A.334)}$$

$$AMT = \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.335)}$$

$$AMU = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.336)}$$

$$AMV = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.337)}$$

$$AMW = \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.338)}$$

$$AMX = \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.339)}$$

$$AMY = \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \quad \text{(A.340)}$$

$$AMZ = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.341)}$$

$$ANA = 2 * \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.342)}$$

$$ANB = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.343)}$$

$$ANC = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi_v)^3 * \sin(\psi) \quad \text{(A.344)}$$

$$AND = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) \quad \text{(A.345)}$$

$$ANE = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.346)}$$

$$ANF = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.347)}$$

$$ANG = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v)^2 \quad \text{(A.348)}$$

$$ANH = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.349)}$$

$$ANI = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v)^3 * \sin(\psi) \quad \text{(A.350)}$$

$$ANJ = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^3 * \sin(\psi) \quad \text{(A.351)}$$

$$ANK = \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta_v) * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.352)}$$

$$ANL = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \quad \text{(A.353)}$$

$$ANM = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.354)}$$

$$ANN = \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^3 * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.355)}$$

$$ANO = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.356)}$$

$$ANP = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.357)}$$

$$ANQ = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^3 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.358)}$$

$$ANR = 2 * \cos(\phi) * \cos(\psi) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \quad \text{(A.359)}$$

$$ANS = \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.360)}$$

$$ANT = \cos(\theta) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 \quad \text{(A.361)}$$

$$ANU = \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v)^2 \quad \text{(A.362)}$$

$$ANV = \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v)^2 \quad \text{(A.363)}$$

$$ANW = \cos(\theta_v) * \cos(\phi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi)^2 \quad \text{(A.364)}$$

$$ANX = \cos(\theta_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^3 * \sin(\psi) \quad \text{(A.365)}$$

$$ANY = \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.366)}$$

$$ANZ = \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.367)}$$

$$AOA = \cos(\theta) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.368)}$$

$$AOB = \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \quad \text{(A.369)}$$

$$AOC = \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.370)}$$

$$AOD = \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.371)}$$

$$AOE = \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \quad \text{(A.372)}$$

$$AOF = \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.373)}$$

$$AOG = \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^3 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.374)}$$

$$AOH = \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) \quad \text{(A.375)}$$

$$AOI = \cos(\theta)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.376)}$$

$$AOJ = \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.377)}$$

$$AOK = \cos(\theta) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v)^2 \quad \text{(A.378)}$$

$$AOL = \cos(\theta_v) * \cos(\phi) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi)^2 * \sin(\psi_v)^2 \quad \text{(A.379)}$$

$$AOM = \cos(\theta_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.380)}$$

$$AON = \cos(\theta) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \quad \text{(A.381)}$$

$$AOO = \cos(\phi) * \cos(\psi) * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \quad \text{(A.382)}$$

$$AOP = \cos(\phi) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.383)}$$

$$AOQ = \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \quad \text{(A.384)}$$

$$AOR = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 \quad \text{(A.385)}$$

$$AOS = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) \quad \text{(A.386)}$$

$$AOT = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^2 \quad \text{(A.387)}$$

$$AOU = 2 * \cos(\theta)^2 * \cos(\theta_v)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.388)}$$

$$AOV = 2 * \cos(\theta)^3 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \quad \text{(A.389)}$$

$$AOW = 3 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.390)}$$

$$AOX = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \quad \text{(A.391)}$$

$$AOY = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \quad \text{(A.392)}$$

$$AOZ = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \quad \text{(A.393)}$$

$$APA = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\psi) * \sin(\psi_v) \quad \text{(A.394)}$$

$$APB = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 \quad \text{(A.395)}$$

$$APC = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi_v)^2 \quad \text{(A.396)}$$

$$APD = 3 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi_v) \tag{A.397}$$

$$APE = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) \tag{A.398}$$

$$APF = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.399}$$

$$APG = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.400}$$

$$APH = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \tag{A.401}$$

$$API = 2 * \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \tag{A.402}$$

$$APJ = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^2 \tag{A.403}$$

$$APK = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.404}$$

$$APL = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi)^2 \tag{A.405}$$

$$APM = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \tag{A.406}$$

$$APN = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \tag{A.407}$$

$$APO = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.408}$$

$$APP = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.409}$$

$$APQ = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^3 * \sin(\phi) * \sin(\psi_v) \tag{A.410}$$

$$APR = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.411}$$

$$APS = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \tag{A.412}$$

$$APT = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi_v) \tag{A.413}$$

$$APU = 3 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.414}$$

$$APV = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.415}$$

$$APW = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.416}$$

$$APX = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \tag{A.417}$$

$$APY = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \tag{A.418}$$

$$APZ = 2 * \cos(\theta) * \cos(\theta_v)^3 * \cos(\phi)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.419}$$

$$AQA = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.420}$$

$$AQB = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^3 \tag{A.421}$$

$$AQC = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^3 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.422}$$

$$AQD = 2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.423}$$

$$AQE = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.424}$$

$$AQF = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2 \tag{A.425}$$

$$AQG = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.426}$$

$$AQH = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.427}$$

$$AQI = 2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.428}$$

$$AQJ = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.429}$$

$$AQK = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi_v)^2 \tag{A.430}$$

$$AQL = 2 * \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \tag{A.431}$$

$$AQM = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.432}$$

$$AQN = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 \tag{A.433}$$

$$AQO = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.434}$$

$$AQP = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.435}$$

$$AQQ = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\psi)^3 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.436}$$

$$AQR = 3 * \cos(\theta) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.437}$$

$$AQS = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.438}$$

$$AQT = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.439}$$

$$AQU = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \tag{A.440}$$

$$AQV = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.441}$$

$$AQW = 2 * \cos(\theta) * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.442}$$

$$AQX = 2 * \cos(\theta) * \cos(\theta_v)^3 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.443}$$

$$AQY = 2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.444}$$

$$AQZ = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v)^3 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \tag{A.445}$$

$$ARA = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.446}$$

$$ARB = 2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.447}$$

$$ARC = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.448}$$

$$ARD = 3 * \cos(\theta) * \cos(\phi) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.449}$$

$$ARE = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) \tag{A.450}$$

$$ARF = 2 * \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \tag{A.451}$$

$$ARG = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.452}$$

$$ARH = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) \tag{A.453}$$

$$ARI = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\psi) * \sin(\psi_v) \tag{A.454}$$

$$ARJ = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.455}$$

$$ARK = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.456}$$

$$ARL = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.457}$$

$$ARM = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.458}$$

$$ARN = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \tag{A.459}$$

$$ARO = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \tag{A.460}$$

$$ARP = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \tag{A.461}$$

$$ARQ = 4 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.462}$$

$$ARR = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \tag{A.463}$$

$$ARS = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \tag{A.464}$$

$$ART = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.465}$$

$$ARU = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.466}$$

$$ARV = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.467}$$

$$ARW = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.468}$$

$$ARX = 2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \tag{A.469}$$

$$ARY = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.470}$$

$$ARZ = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.471}$$

$$ASA = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.472}$$

$$ASB = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.473}$$

$$ASC = 2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi_v) \tag{A.474}$$

$$ASD = 4 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.475}$$

$$ASE = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi)^2 \tag{A.476}$$

$$ASF = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.477}$$

$$ASG = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \tag{A.478}$$

$$ASH = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \tag{A.479}$$

$$ASI = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.480}$$

$$ASJ = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \tag{A.481}$$

$$ASK = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.482}$$

$$ASL = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.483}$$

$$ASM = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.484}$$

$$ASN = 2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.485}$$

$$ASO = 2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.486}$$

$$ASP = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.487}$$

$$ASQ = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.488}$$

$$ASR = 2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.489}$$

$$ASS = 2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.490}$$

$$AST = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.491}$$

$$ASU = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \sin(\theta) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.492}$$

$$ASV = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2 \tag{A.493}$$

$$ASW = 2 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^3 * \sin(\psi_v) \tag{A.494}$$

$$ASX = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v)^2 \tag{A.495}$$

$$ASY = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.496}$$

$$ASZ = 2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.497}$$

$$ATA = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \tag{A.498}$$

$$ATB = 2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.499}$$

$$ATC = 2 * \cos(\theta) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^3 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi) * \sin(\psi_v) \tag{A.500}$$

$$ATD = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v)^2 * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.501}$$

$$ATE = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.502}$$

$$ATF = 2 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\psi) \tag{A.503}$$

$$ATG = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi) \tag{A.504}$$

$$ATH = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \tag{A.505}$$

$$ATI = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \tag{A.506}$$

$$ATJ = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi_v) * \sin(\psi_v) \tag{A.507}$$

$$ATK = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi_v) \tag{A.508}$$

$$ATL = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \tag{A.509}$$

$$ATM = 3 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \tag{A.510}$$

$$ATN = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \tag{A.511}$$

$$ATO = 3 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) \tag{A.512}$$

$$ATP = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.513}$$

$$ATQ = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \tag{A.514}$$

$$ATR = 2 * \cos(\theta)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi) \tag{A.515}$$

$$ATS = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.516}$$

$$ATT = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.517}$$

$$ATU = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi) * \sin(\psi_v)^2 \tag{A.518}$$

$$ATV = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi) \tag{A.519}$$

$$ATW = 2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi_v) * \sin(\psi) \tag{A.520}$$

$$ATX = 2 * \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.521}$$

$$ATY = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.522}$$

$$ATZ = 2 * \cos(\theta) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.523}$$

$$AUA = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \tag{A.524}$$

$$AUB = 2 * \cos(\theta)^3 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.525}$$

$$AUC = 3 * \cos(\theta) * \cos(\theta_v)^2 * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.526}$$

$$AUD = 2 * \cos(\theta) * \cos(\phi) * \cos(\phi_v)^2 * \cos(\psi) * \cos(\psi_v) * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.527}$$

$$AUE = 2 * \cos(\theta) * \cos(\theta_v)^3 * \cos(\phi) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.528}$$

$$AUF = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v) \tag{A.529}$$

$$AUG = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^3 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^2 \tag{A.530}$$

$$AUH = 3 * \cos(\theta) * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.531}$$

$$AUI = 3 * \cos(\theta) * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \sin(\theta) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.532}$$

$$AUJ = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi) * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v) \tag{A.533}$$

$$AUK = 2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi) \tag{A.534}$$

$$AUL = 2 * \cos(\theta) * \cos(\phi) * \cos(\psi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v) * \sin(\phi) * \sin(\phi_v)^2 * \sin(\psi_v) \tag{A.535}$$

$$AUM = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi) * \sin(\phi_v)^3 * \sin(\psi) * \sin(\psi_v) \tag{A.536}$$

$$AUN = 2 * \cos(\theta) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v) * \sin(\psi_v) \tag{A.537}$$

$$AUO = 2 * \cos(\theta_v) * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\psi)^2 * \sin(\psi_v) \tag{A.538}$$

$$AUP = 2 * \cos(\theta) * \cos(\theta_v) * \cos(\phi) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi)^3 * \sin(\psi_v) \tag{A.539}$$

$$AUQ = 2 * \cos(\theta)^3 * \cos(\phi) * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi) * \sin(\phi_v) * \sin(\psi) * \sin(\psi_v) \tag{A.540}$$

$$AUR = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\psi)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.541}$$

$$AUS = 4*\cos(\theta)*\cos(\theta_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)^2*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v) \tag{A.542}$$

$$AUT = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\psi_v) \tag{A.543}$$

$$AUU = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)^2*\sin(\theta)*\sin(\theta_v)*\sin(\phi_v)^2*\sin(\psi)^2 \tag{A.544}$$

$$AUV = 3*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)^2*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2 \tag{A.545}$$

$$AUW = 2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)^3*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.546}$$

$$AUX = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi_v)^2*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2 \tag{A.547}$$

$$AUY = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)^2*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^3 \tag{A.548}$$

$$AUZ = 3*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta_v)^2*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.549}$$

$$AVA = 2*\cos(\theta_v)*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^3*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.550}$$

$$AVB = 2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^3*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.551}$$

$$AVC = 2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^3*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.552}$$

$$AVD = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)^2*\sin(\theta)*\sin(\theta_v)*\sin(\phi)^2*\sin(\phi_v)^2*\sin(\psi)^2 \tag{A.553}$$

$$AVE = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)*\cos(\psi)*\sin(\theta)*\sin(\theta_v)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v)^2 \tag{A.554}$$

$$AVF = 3*\cos(\theta)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)^2*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2 \tag{A.555}$$

$$AVG = 3*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v)^2 \tag{A.556}$$

$$AVH = 4*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)^2*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.557}$$

$$AVI = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi_v)^3*\sin(\psi)^2*\sin(\psi_v) \tag{A.558}$$

$$AVJ = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v)^2 \tag{A.559}$$

$$AVK = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^3*\sin(\psi_v)^2 \tag{A.560}$$

$$AVL = 3*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.561}$$

$$AVM = 3*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta_v)^2*\sin(\phi)^2*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.562}$$

$$AVN = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)^2*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v)^2 \tag{A.563}$$

$$AVO = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)^3*\sin(\psi)^3*\sin(\psi_v) \tag{A.564}$$

$$AVP = 3*\cos(\theta)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v)^2 \tag{A.565}$$

$$AVQ = 3*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)*\sin(\theta)^3*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.566}$$

$$AVR = 4*\cos(\theta)*\cos(\theta_v)^2*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)^2*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.567}$$

$$AVS = 3*\cos(\theta)*\cos(\theta_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)^2*\sin(\phi_v)^3*\sin(\psi)^2*\sin(\psi_v) \tag{A.568}$$

$$AVT = 3*\cos(\theta)*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^3*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.569}$$

$$AVU = 3*\cos(\theta_v)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)^2*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v) \tag{A.570}$$

$$AVV = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2 \tag{A.571}$$

$$AVW = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^2 \tag{A.572}$$

$$AVX = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi)^2*\cos(\psi_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.573}$$

$$AVY = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi_v)*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.574}$$

$$AVZ = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.575}$$

$$AWA = 4*\cos(\theta)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.576}$$

$$AWB = 2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^3*\sin(\theta_v)^3*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.577}$$

$$AWC = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\cos(\psi_v)^2*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\psi) \tag{A.578}$$

$$AWD = 4*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\psi_v) \tag{A.579}$$

$$AWE = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)*\sin(\psi_v) \tag{A.580}$$

$$AWF = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi_v)*\sin(\psi_v) \tag{A.581}$$

$$AWG = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\cos(\psi_v)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi) \tag{A.582}$$

$$AWH = 2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.583}$$

$$AWI = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v) \tag{A.584}$$

$$AWJ = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)^2*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi) \tag{A.585}$$

$$AWK = 2*\cos(\theta_v)*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.586}$$

$$AWL = 2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.587}$$

$$AWM = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\psi)*\sin(\psi_v)^2 \tag{A.588}$$

$$AWN = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)^2*\sin(\phi_v)*\sin(\psi_v) \tag{A.589}$$

$$AWO = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v)^2 \tag{A.590}$$

$$AWP = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)*\sin(\theta)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.591}$$

$$AWQ = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi_v)*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)^3*\sin(\psi_v) \tag{A.592}$$

$$AWR = 3*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi_v) \tag{A.593}$$

$$AWS = 3*\cos(\theta)^3*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi_v)*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.594}$$

$$AWT = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v) \tag{A.595}$$

$$AWU = 3*\cos(\theta)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\psi)^2*\sin(\psi_v) \tag{A.596}$$

$$AWV = 3*\cos(\theta)*\cos(\theta_v)^3*\cos(\phi)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.597}$$

$$AWW = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi_v)^2*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\phi)^2*\sin(\phi_v)*\sin(\psi)^2*\sin(\psi_v) \tag{A.598}$$

$$AWX = 4*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.599}$$

$$AWY = 3*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)^3*\sin(\psi)*\sin(\psi_v) \tag{A.600}$$

$$AWZ = 4*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)*\sin(\psi_v)^2 \tag{A.601}$$

$$AXA = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.602}$$

$$AXB = 4*\cos(\theta)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.603}$$

$$AXC = 4*\cos(\theta_v)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.604}$$

$$AXD = 6*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.605}$$

$$AXE = 6*\cos(\theta)*\cos(\theta_v)^2*\cos(\phi)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)^2*\sin(\psi)^2*\sin(\psi_v) \tag{A.606}$$

$$AXF = 4*\cos(\theta)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)*\sin(\theta_v)^3*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.607}$$

$$AXG = 4*\cos(\theta_v)^2*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\cos(\psi_v)*\sin(\theta)^3*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.608}$$

$$AXH = 6*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)^2*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.609}$$

$$AXI = 6*\cos(\theta)*\cos(\theta_v)*\cos(\phi)*\cos(\psi)^2*\cos(\psi_v)*\sin(\theta)^2*\sin(\theta_v)^2*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.610}$$

$$AXJ = 4*\cos(\theta)^2*\cos(\theta_v)*\cos(\phi)*\cos(\phi_v)*\cos(\psi)*\sin(\theta)*\sin(\theta_v)*\sin(\phi)*\sin(\phi_v)*\sin(\psi)*\sin(\psi_v) \tag{A.611}$$

The determinant of H_k is now seen in the following equation.

$$
\begin{aligned}
\det(H_k) = &- \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi)^2 \\
&- \cos(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&- \cos(\phi)^2 * \sin(\theta)^2 * \sin(\psi)^2 \\
&- \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) \\
&+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) \\
&+ AAA \\
&+ \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi)^2 \\
&+ \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta)^2 * \sin(\psi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \sin(\phi)^2 * \sin(\psi)^2 \\
&+ \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi)^2 \\
&+ \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi)^2 \\
&+ \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi)^2 \\
&+ \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi)^2 \\
&+ AAB + AAC - AAD + AAE \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi_v)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\phi)^2 * \sin(\psi_v)^2
\end{aligned}
$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\psi)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\psi)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2$$

$$+ \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2$$

$$- AAF - AAG - AAH + AAI + AAJ$$

$$+ AAK - AAL$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi_v)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\psi)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\phi)^2 * \cos(\phi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$+ \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)$$

$$- \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)$$

$$- AAM - AAN - AAO - AAP - AAQ$$

$$- AAR - AAS$$

$$+ \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\psi)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi_v)^2$$

$$- \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\phi)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi)^2 * \sin(\phi_v)^2$$

$$- \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\psi)^2$$

$$
\begin{aligned}
&- \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\psi)^2 \\
&- \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 \\
&- \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\psi)^2 \\
&- \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 \\
&+ AAT + AAU + AAV + AAW \\
&- \cos(\theta_v) * \cos(\phi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2 \\
&- \cos(\theta_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 \\
&- \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&- AAX - AAY - AAZ + ABA - ABB \\
&+ ABC + ABD + ABE + ABF + ABG \\
&- ABH + ABI - ABJ - ABK \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 \\
&+ ABL + ABM - ABN - ABO - ABP \\
&+ ABQ + ABR + ABS + ABT - ABU \\
&+ ABV + ABW + ABX - ABY + ABZ \\
&- ACA - ACB + ACC + ACD - ACE \\
&- ACF - ACG - ACH + ACI - ACJ \\
&- ACK - ACL + ACM + ACN - ACO \\
&- ACP - ACQ - ACR + ACS - ACT \\
&+ ACU - ACV - ACW + ACX - ACY \\
&- ACZ - ADA + ADB + ADC + ADD \\
&+ ADE - ADF - ADG - ADH + ADI \\
&- ADJ + ADK - ADL + ADM - ADN \\
&- ADO - ADP - ADQ - ADR - ADS
\end{aligned}
$$

$$+ ADT - ADU - ADV + ADW + ADX$$
$$+ ADY - ADZ + AEA - AEB + AEC$$
$$- AED - AEE - AEF - AEG + AEH$$
$$+ AEI + AEJ + AEK + AEL + AEM$$
$$+ AEN + AEO - AEP + AEQ - AER$$
$$+ AES - AET + AEU + AEV - AEW$$
$$- AEX - AEY - AEZ + AFA - AFB$$
$$+ AFC - AFD + AFE - AFF + AFG$$
$$+ AFH + AFI - AFJ - AFK + AFL$$
$$- AFM - AFN - AFO + AFP + AFQ$$
$$- AFR - AFS - AFT - AFU + AFV$$
$$- AFW + AFX + AFY + AFZ - AGA$$
$$- AGB - AGC + AGD - AGE + AGF$$
$$+ AGG - AGH - AGI - AGJ + AGK$$
$$- AGL + AGM - AGN + AGO - AGP$$
$$- AGQ + AGR - AGS - AGT - AGU$$
$$- AGV + AGW + AGX - AGY - AGZ$$
$$+ AHA + AHB + AHC + AHD + AHE$$
$$+ AHF + AHG + AHH + AHI - AHJ$$
$$- AHK - AHL - AHM - AHN - AHO$$
$$+ AHP + AHQ + AHR + AHS - AHT$$
$$- AHU - AHV - AHW + AHX - AHY$$
$$+ AHZ + AIA + AIB + AIC - AID$$
$$- AIE - AIF + AIG - AIH - AII$$

$$
\begin{aligned}
&- AIJ - AIK - AIL - AIM - AIN \\
&- AIO + AIP + AIQ + AIR + AIS \\
&+ AIT + AIU + AIV + AIW + AIX \\
&+ AIY - AIZ - AJA - AJB + AJC \\
&+ AJD + AJE + AJF - AJG - AJH \\
&+ AJI + AJJ + AJK + AJL + AJM \\
&+ AJN - AJO - AJP - AJQ + AJR \\
&+ AJS + AJT + AJU + AJV + AJW \\
&+ AJX + AJY + AJZ + AKA + AKB \\
&+ AKC + AKD + AKE + AKF - AKG \\
&+ AKH - AKI - AKJ - AKK - AKL \\
&- AKM - AKN - AKO - AKP + AKQ \\
&+ AKR - AKS - AKT - AKU + AKV \\
&- AKW - AKX - AKY - AKZ - ALA \\
&- ALB + ALC + ALD - ALE - ALF \\
&- ALG - ALH - ALI - ALJ - ALK \\
&- ALL - ALM - ALN - ALO - ALP \\
&- ALQ - ALR + ALS + ALT + ALU \\
&- ALV + ALW + ALX + ALY + ALZ \\
&+ AMA + AMB + AMC + AMD + AME \\
&+ AMF + AMG + AMH + AMI + AMJ \\
&+ AMK + AML + AMM + AMN + AMO \\
&- AMP - AMQ - AMR - AMS - AMT \\
&- AMU - AMV - AMW - AMX - AMY
\end{aligned}
$$

$$
\begin{aligned}
&- AMZ - ANA - ANB - ANC - AND \\
&- ANE + ANF + ANG + ANH + ANI \\
&+ ANJ + ANK + ANL + ANM + ANN \\
&+ ANO + ANP + ANQ - ANR - ANS \\
&- ANT - ANU - ANV + ANW - ANX \\
&- ANY - ANZ + AOA - AOB - AOC \\
&- AOD - AOE - AOF - AOG + AOH \\
&+ AOI + AOJ + AOK - AOL + AOM \\
&- AON + AOO + AOP + AOQ + AOR \\
&+ AOS + AOT + AOU + AOV + AOW \\
&+ AOX + AOY - AOZ + APA - APB \\
&- APC - APD - APE - APF + APG \\
&+ APH - API - APJ - APK - APL \\
&- APM + APN - APO - APP + APQ \\
&- APR - APS + APT - APU - APV \\
&- APW + APX + APY - APZ - AQA \\
&- AQB + AQC - AQD - AQE + AQF \\
&+ AQG - AQH - AQI - AQJ - AQK \\
&+ AQL + AQM + AQN - AQO - AQP \\
&+ AQQ + AQR - AQS + AQT - AQU \\
&- AQV - AQW + AQX + AQY - AQZ \\
&+ ARA + ARB + ARC + ARD - ARE \\
&- ARF - ARG - ARH - ARI - ARJ \\
&+ ARK + ARL + ARM + ARN + ARO
\end{aligned}
$$

$$+ ARP + ARQ - ARR - ARS - ART$$
$$- ARU - ARV - ARW - ARX - ARY$$
$$- ARZ - ASA - ASB - ASC + ASD$$
$$- ASE - ASF - ASG + ASH + ASI$$
$$+ ASJ + ASK + ASL + ASM + ASN$$
$$+ ASO + ASP + ASQ + ASR + ASS$$
$$- AST + ASU + ASV + ASW - ASX$$
$$- ASY - ASZ - ATA - ATB + ATC$$
$$- ATD - ATE - ATF - ATG - ATH$$
$$- ATI - ATJ + ATK + ATL + ATM$$
$$+ ATN + ATO + ATP + ATQ + ATR$$
$$+ ATS + ATT + ATU + ATV + ATW$$
$$+ ATX + ATY + ATZ + AUA + AUB$$
$$+ AUC + AUD + AUE + AUF + AUG$$
$$- AUH - AUI - AUJ - AUK - AUL$$
$$+ AUM - AUN - AUO + AUP - AUQ$$
$$- AUR - AUS + AUT + AUU + AUV$$
$$+ AUW + AUX + AUY + AUZ + AVA$$
$$+ AVB + AVC - AVD - AVE - AVF$$
$$- AVG + AVH + AVI - AVJ - AVK$$
$$- AVL - AVM + AVN + AVO + AVP$$
$$- AVQ - AVR - AVS - AVT + AVU$$
$$- AVV - AVW + AVX - AVY - AVZ$$
$$+ AWA + AWB - AWC - AWD - AWE$$

$$
\begin{aligned}
&- AWF - AWG + AWH - AWI + AWJ \\
&+ AWK + AWL + AWM + AWN + AWO \\
&+ AWP - AWQ + AWR + AWS + AWT \\
&+ AWU + AWV + AWW + AWX - AWY \\
&- AWZ - AXA - AXB - AXC + AXD \\
&+ AXE - AXF - AXG - AXH - AXI \\
&- AXJ \\
&+ 8 * \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi) * \cos(\phi_v) * \cos(\psi) * \cos(\psi_v) * \\
&) - \cos(\theta)^2 * \cos(\phi)^2 * \sin(\psi)^2 \\
&- \cos(\theta)^2 * \cos(\psi)^2 * \sin(\phi)^2 \\
&- \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta)^2 \\
&- \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&+ \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\psi)^2 \\
&+ \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\psi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\psi)^2 \\
&+ \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 \\
&- AAA \\
&+ \cos(\theta_v) * \cos(\phi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&+ \cos(\theta_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2
\end{aligned}
$$

$$+ \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$- AAB - AAC + AAD - AAE + AAF$$

$$+ AAG + AAH$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v)^2 * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\psi)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\psi_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \sin(\theta_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\theta)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$+ \cos(\theta)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \sin(\theta)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$+ \cos(\theta_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2$$

$$+ \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi)^2$$

$$+ \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2$$

$$+ \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2$$

$$- AAI - AAJ - AAK + AAL + AAM$$

$$+ AAN$$

$$+ \cos(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\theta_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\phi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$+ \cos(\psi)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi_v)^2$$

$$+ \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\phi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi_v)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\phi)^2$$

$$- \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\psi)^2$$

$$- \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta_v)^2$$

$$- \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2$$

$$+ AAO + AAP + AAQ + AAR + AAS$$

$$- AAT - AAU - AAV - AAW$$

$$- \cos(\theta_v) * \cos(\phi_v) * \cos(\psi)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi_v)^2$$

$$- \cos(\theta_v) * \cos(\phi_v) * \cos(\psi_v)^2 * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2$$

$$- \cos(\theta_v) * \cos(\phi)^2 * \cos(\phi_v) * \sin(\theta)^2 * \sin(\psi)^2 * \sin(\psi_v)^2$$

$$
\begin{aligned}
&- \cos(\theta_v) * \cos(\phi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 \\
&- \cos(\theta_v) * \cos(\phi_v)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&- \cos(\theta_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\phi_v)^2 \\
&- \cos(\phi_v) * \cos(\psi)^2 * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\phi)^2 \\
&- \cos(\theta)^2 * \cos(\theta_v) * \cos(\phi_v) * \sin(\phi)^2 * \sin(\psi)^2 * \sin(\psi_v)^2 \\
&- \cos(\theta)^2 * \cos(\theta_v) * \cos(\psi_v) * \sin(\phi)^2 * \sin(\phi_v)^2 * \sin(\psi)^2 \\
&- \cos(\theta)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta_v)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&- \cos(\theta_v)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\phi)^2 * \sin(\psi)^2 \\
&- \cos(\phi)^2 * \cos(\phi_v) * \cos(\psi_v) * \sin(\theta)^2 * \sin(\theta_v)^2 * \sin(\psi)^2 \\
&+ AAX + AAY + AAZ - ABA + ABB \\
&- ABC - ABD - ABE - ABF - ABG \\
&+ ABH - ABI + ABJ + ABK \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \sin(\psi_v)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi_v)^2 * \sin(\psi)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\phi_v)^2 \\
&+ \cos(\theta)^2 * \cos(\theta_v)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\phi)^2 \\
&+ \cos(\theta)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta_v)^2 \\
&+ \cos(\theta_v)^2 * \cos(\phi)^2 * \cos(\phi_v)^2 * \cos(\psi)^2 * \cos(\psi_v)^2 * \sin(\theta)^2 \\
&- ABL - ABM + ABN + ABO + ABP \\
&- ABQ - ABR - ABS - ABT + ABU \\
&- ABV - ABW - ABX + ABY - ABZ \\
&+ ACA + ACB - ACC - ACD + ACE \\
&+ ACF + ACG + ACH - ACI + ACJ \\
&+ ACK + ACL - ACM - ACN + ACO
\end{aligned}
$$

$$+ ACP + ACQ + ACR - ACS + ACT$$
$$- ACU + ACV + ACW - ACX + ACY$$
$$+ ACZ + ADA - ADB - ADC - ADD$$
$$- ADE + ADF + ADG + ADH - ADI$$
$$+ ADJ - ADK + ADL - ADM + ADN$$
$$+ ADO + ADP + ADQ + ADR + ADS$$
$$- ADT + ADU + ADV - ADW - ADX$$
$$- ADY + ADZ - AEA + AEB - AEC$$
$$+ AED + AEE + AEF + AEG - AEH$$
$$- AEI - AEJ - AEK - AEL - AEM$$
$$- AEN - AEO + AEP - AEQ + AER$$
$$- AES + AET - AEU - AEV + AEW$$
$$+ AEX + AEY + AEZ - AFA + AFB$$
$$- AFC + AFD - AFE + AFF - AFG$$
$$- AFH - AFI + AFJ + AFK - AFL$$
$$+ AFM + AFN + AFO - AFP - AFQ$$
$$+ AFR + AFS + AFT + AFU - AFV$$
$$+ AFW - AFX - AFY - AFZ + AGA$$
$$+ AGB + AGC - AGD + AGE - AGF$$
$$- AGG + AGH + AGI + AGJ - AGK$$
$$+ AGL - AGM + AGN - AGO + AGP$$
$$+ AGQ - AGR + AGS + AGT + AGU$$
$$+ AGV - AGW - AGX + AGY + AGZ$$
$$- AHA - AHB - AHC - AHD - AHE$$

$$
\begin{aligned}
&- AHF - AHG - AHH - AHI + AHJ \\
&+ AHK + AHL + AHM + AHN + AHO \\
&- AHP - AHQ - AHR - AHS + AHT \\
&+ AHU + AHV + AHW - AHX + AHY \\
&- AHZ - AIA - AIB - AIC + AID \\
&+ AIE + AIF - AIG + AIH + AII \\
&+ AIJ + AIK + AIL + AIM + AIN \\
&+ AIO - AIP - AIQ - AIR - AIS \\
&- AIT - AIU - AIV - AIW - AIX \\
&- AIY + AIZ + AJA + AJB - AJC \\
&- AJD - AJE - AJF + AJG + AJH \\
&- AJI - AJJ - AJK - AJL - AJM \\
&- AJN + AJO + AJP + AJQ - AJR \\
&- AJS - AJT - AJU - AJV - AJW \\
&- AJX - AJY - AJZ - AKA - AKB \\
&- AKC - AKD - AKE - AKF + AKG \\
&- AKH + AKI + AKJ + AKK + AKL \\
&+ AKM + AKN + AKO + AKP - AKQ \\
&- AKR + AKS + AKT + AKU - AKV \\
&+ AKW + AKX + AKY + AKZ + ALA \\
&+ ALB - ALC - ALD + ALE + ALF \\
&+ ALG + ALH + ALI + ALJ + ALK \\
&+ ALL + ALM + ALN + ALO + ALP \\
&+ ALQ + ALR - ALS - ALT - ALU
\end{aligned}
$$

$$+ ALV - ALW - ALX - ALY - ALZ$$
$$- AMA - AMB - AMC - AMD - AME$$
$$- AMF - AMG - AMH - AMI - AMJ$$
$$- AMK - AML - AMM - AMN - AMO$$
$$+ AMP + AMQ + AMR + AMS + AMT$$
$$+ AMU + AMV + AMW + AMX + AMY$$
$$+ AMZ + ANA + ANB + ANC + AND$$
$$+ ANE - ANF - ANG - ANH - ANI$$
$$- ANJ - ANK - ANL - ANM - ANN$$
$$- ANO - ANP - ANQ + ANR + ANS$$
$$+ ANT + ANU + ANV - ANW + ANX$$
$$+ ANY + ANZ - AOA + AOB + AOC$$
$$+ AOD + AOE + AOF + AOG - AOH$$
$$- AOI - AOJ - AOK + AOL - AOM$$
$$+ AON - AOO - AOP - AOQ - AOR$$
$$- AOS - AOT - AOU - AOV - AOW$$
$$- AOX - AOY + AOZ - APA + APB$$
$$+ APC + APD + APE + APF - APG$$
$$- APH + API + APJ + APK + APL$$
$$+ APM - APN + APO + APP - APQ$$
$$+ APR + APS - APT + APU + APV$$
$$+ APW - APX - APY + APZ + AQA$$
$$+ AQB - AQC + AQD + AQE - AQF$$
$$- AQG + AQH + AQI + AQJ + AQK$$

$$
\begin{aligned}
&- AQL - AQM - AQN + AQO + AQP \\
&- AQQ - AQR + AQS - AQT + AQU \\
&+ AQV + AQW - AQX - AQY + AQZ \\
&- ARA - ARB - ARC - ARD + ARE \\
&+ ARF + ARG + ARH + ARI + ARJ \\
&- ARK - ARL - ARM - ARN - ARO \\
&- ARP - ARQ + ARR + ARS + ART \\
&+ ARU + ARV + ARW + ARX + ARY \\
&+ ARZ + ASA + ASB + ASC - ASD \\
&+ ASE + ASF + ASG - ASH - ASI \\
&- ASJ - ASK - ASL - ASM - ASN \\
&- ASO - ASP - ASQ - ASR - ASS \\
&+ AST - ASU - ASV - ASW + ASX \\
&+ ASY + ASZ + ATA + ATB - ATC \\
&+ ATD + ATE + ATF + ATG + ATH \\
&+ ATI + ATJ - ATK - ATL - ATM \\
&- ATN - ATO - ATP - ATQ - ATR \\
&- ATS - ATT - ATU - ATV - ATW \\
&- ATX - ATY - ATZ - AUA - AUB \\
&- AUC - AUD - AUE - AUF - AUG \\
&+ AUH + AUI + AUJ + AUK + AUL \\
&- AUM + AUN + AUO - AUP + AUQ \\
&+ AUR + AUS - AUT - AUU - AUV \\
&- AUW - AUX - AUY - AUZ - AVA
\end{aligned}
$$

$$
\begin{aligned}
&- AVB - AVC + AVD + AVE + AVF \\
&+ AVG - AVH - AVI + AVJ + AVK \\
&+ AVL + AVM - AVN - AVO - AVP \\
&+ AVQ + AVR + AVS + AVT - AVU \\
&+ AVV + AVW - AVX + AVY + AVZ \\
&- AWA - AWB + AWC + AWD + AWE \\
&+ AWF + AWG - AWH + AWI - AWJ \\
&- AWK - AWL - AWM - AWN - AWO \\
&- AWP + AWQ - AWR - AWS - AWT \\
&- AWU - AWV - AWW - AWX + AWY \\
&+ AWZ + AXA + AXB + AXC - AXD \\
&- AXE + AXF + AXG + AXH + AXI
\end{aligned}
$$

Note that every previously defined constant has a corresponding constant of opposite sign. By removing like terms and leveraging the identity $\sin(x)^2 + \cos(x)^2 = 1$ to simplify, the determinant reduces to zero. Hence using H_k as defined in Equation (3.9), Equation A.612 is obtained for a single measurement.

$$\det(H_k) = 0 \tag{A.612}$$

Therefore this shows that a single measurement lacks sufficient information for H_k to be observable.