Automation of the multi-sensor system calibration for mobile robotic applications

Automatyzacja procesu kalibracji systemu wielosensorycznego dla aplikacji robotyki mobilnej

by

Michał PEŁKA

Supervisor

Janusz BĘDKOWSKI, Ph.D., D.Sc,

Faculty of Electrical Engineering
Poznań University of Technology

2022

Abstract

Automation of the multi-sensor system calibration for mobile robotic applications

Michał PEŁKA

This dissertation addresses problems encountered in development of new robotic mapping systems. It is a broad and interesting topic involving many fields, such as system design, mechatronics, software development and requires interdisciplinary knowledge. In the research concluded in this work the author focuses on methods of automation calibration for various robotic mapping systems. What is essential in building optimization problems are the chosen domain of the sensors, methods of calibration, Field Of View (FOV), and their overlap. That was shown in work both from a theoretical point of view and practical applications. First, existing knowledge regarding Simultaneous Localisation and Mapping (SLAM), on-manifold optimization was reviewed. Second, the methodology for automated calibration was presented. Finally, multiple systems were presented with a detailed description of the method used and their results. Shown approaches and methods are valuable tools in many fields such as mobile mapping system design, robotic system design, or autonomous driving perception.

Abstrakt

Automatyzacja procesu kalibracji systemu wielosensorycznego dla aplikacji robotyki mobilnej

Michał PEŁKA

Niniejsza rozprawa dotyczy problemów napotykanych podczas opracowywania nowych robotycznych systemów mapowania. Jest to obszerny i interesujący temat obejmujący wiele dziedzin, takich jak projektowanie złożonych systemów, mechatronika, tworzenie oprogramowania i wymaga od projektanta wykazania się wiedzą interdyscyplinarną. W badaniach zakończonych niniejszą pracą autor skupił się na metodach zautomatyzowania kalibracji różnych robotycznych systemów mapowania. Wybrana technologia czujników, sposób kalibracji, kształt pola widzenia i ich wzajemne relacje geometryczne są aspektami, które są kluczowe w budowaniu problemów optymalizacyjnych. W pracy zostało to pokazane zarówno z teoretycznego punktu widzenia, jak i zastosowań praktycznych. Przeanalizowano istniejącą wiedzę na temat projektowania SLAM, optymalizacji z wykorzystaniem algebry Liego. Po czym sformułowano metodykę dotyczącą automatycznej kalibracji. Następnie przedstawiono kilka systemów ze szczegółowym opisem zastosowanych metod i analizą wyników. Przedstawione podejścia i metodyka są cennymi narzędziami w wielu dziedzinach, takich jak projektowanie systemów mapowania mobilnego, projektowanie systemów robotycznych czy rozwój technologii pojazdów autonomicznych.

Acknowledgements

I would like to express my sincere gratitude to my supervisor D.Sc. Janusz Będkowski for the continuous support of my Ph.D. study and research. This work was partially supported by the following projects: NCBiR (Polish National Center for Research and Development) project: "Research of Mobile Spatial Assistance System" Nr: LIDER/036/659/L-4/12/NCBR/2013; European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n° 285417 ICARUS project - Integrated Components for Assisted Rescue and Unmanned Search operations; European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n° 284747 TIRAMISU project - Toolbox Implementation for Removal of Anti-personnel Mines, Submunitions and UXO; NCBiR(Polish National Center for Research and Development) PBS3/B3/25/2015: "Badanie nowej głowicy skanowania przestrzennego oraz jej modyfikacji z przeznaczeniem dla robotów mobilnych do mapowania otoczenia";

Contents

Abstract **i**

Acknowledgements **iii**

List of Figures **vii**

List of Tables **xiv**

List of Abbreviations **xv**

1 Introduction **1**
- 1.1 Thesis outline . . . 2
- 1.2 Sensors in ground robotic mapping systems . . . 2
  - 1.2.1 LiDAR . . . 2
  - 1.2.2 Cameras . . . 3
    - Spherical cameras . . . 4
    - Implementation of spherical camera . . . 5
  - 1.2.3 IMU . . . 10
  - 1.2.4 Odometry . . . 12
  - 1.2.5 GNSS . . . 12
    - RTK and DGPS . . . 13
    - GNSS+INS . . . 13
- 1.3 Map representation . . . 14
  - 1.3.1 Dense representation . . . 14
    - Occupancy grid map . . . 14
    - Elevation map . . . 14
  - 1.3.2 Sparse representation . . . 14
    - Topological map . . . 14
    - Semantic map . . . 15
  - 1.3.3 High Definition map . . . 15

2 Problem formulation **17**

3 Methodology **20**
- 3.1 Time synchronization . . . 20
    - Hardware time synchronization . . . 20
- 3.2 Trajectory . . . 21
  - 3.2.1 Rotation representation . . . 21
    - Groups . . . 21
    - Lie Groups . . . 22

Special orthogonality group $SO(3)$ 24
Lie algebra $\mathfrak{so}(3)$ 25
Rodrigues' rotation formula for rotation matrix 26
Quaternions 27
Quaternion interpolation 32
Proper Euler angles and Tait-Bryan angles 34
Rigid body transformations in $SE(3)$ and their Lie algebra 35
3.2.2 Motion model and smoothness 35
Relative Pose 35
Smoothness 36
3.3 SLAM 36
3.3.1 Factor graph 38
3.3.2 Graph SLAM 39
3.3.3 Bundle Adjustment 40
3.3.4 Robust Nonlinear Least Squares 41
3.3.5 Map and trajectory accuracy assessment 45
3.3.6 Ground truth data sources 46
Synthetic data sources 46
Real data 48
3.4 LiDAR observation equations 49
3.4.1 Point to Point 49
Jacobian matrix in SE(3) for point transformation 50
Iterative Closest Point 51
3.5 Camera observation equations 54
3.5.1 Line to line in equirectangular image 54
Curve fitting 58
Extracting Plücker line from a point cloud 58
Extrinsic calibration of a spherical camera 59
3.5.2 Line to line in rectilinear image 60

4 Experimental validation **63**
4.1 Calibration of a high-volume mobile 3D scanner 63
Modeling a mechanical design as a factor graph 66
4.2 Calibration of multi planar reflectors 73
Mechanical design 73
Geometry of a reflected ray 76
Calibration procedure 78
4.3 Calibration of a rotated reflector 83
Time synchronisation with a Livox Light Detection and Ranging (LiDAR) 83
Results 89
4.4 Calibration of a lightweight 3D unit 89
Mechatronic design 89
Calibration pattern 91
Calibration pattern - LiDAR's intensity domain 93
Cost function 94
Results 95

4.5 Calibration of a mobile backpack mapping system 97
Optimization of trajectory 97
Optimization of an extrinsic laser scanner calibration 100
Quantitative and qualitative validation of extrinsic laser scanner calibration 101
Optimization of an extrinsic calibration of the spherical camera against a laser scanner 103
4.6 Calibration of a 3D unit 107
Mechatronical design 107
Calibration 107
4.6.1 Comparison of the equirectangular and perspective model 108

5 Robotic applications 113
5.1 Mining shaft mapping 113
Calibration method 114
Accuracy assessment 118
5.2 Nuclear Power Plant Mapping 120
5.3 Mobile robot localization 125
Localization using sequential Monte-Carlo 125
Map building and trajectory ground truth 125
Localization algorithm 127
Results 129

6 Conclusions 131
6.1 Thesis contributions 132
6.2 Further research directions 133

Bibliography 134

List of Figures

1.1 Coordinate system used in equirectangular image. 5
1.2 Simulated spherical camera rig with t = 0.05m. 7
1.3 Panorama rendered with an ideal equirectangular camera. 7
1.4 Panorama rendered with six camera rig from figure 1.2, projected on sphere with radius 2 meters. Projected field of view boundaries are marked in color. 8
1.5 Panorama rendered with six camera rig from figure 1.2, projected on sphere with radius 10 meters. Projected field of view boundaries are marked in color. 8
1.6 Local comparison of stitching errors. Left column is ground truth obtained with the ideal equirectangular camera simulation. Next columns are projections for sphere size: 1, 5, 10, 100 meters. Note that optimal radius is different for those two patches. 9
1.7 Parallax error 'X'. Point P is projected to a point P_s' in an idealized spherical model. In real camera point P is projected to P_c'. That is caused by a fact, that projection center for spherical image is Os and for perspective camera is Oc. Those are t apart. 9
1.8 The value of parallax error in meters for given camera $\varepsilon = 60^\circ$ and $t = 0.1m$. 10

2.1 Examples of commercially available Mobile Mapping System (MMS)s. . 18

3.1 The example of applying non-valid ($\det(\boldsymbol{R}) = 1.5$) rotation matrix. Left: original object, middle: proper rotation, right: non-SO3 quasi-rotation. It is evident that the object is scaled. 25
3.2 An example of a left multiplication of quaternion $[0, j]$ in a 3D projection. A quaternion is a four-dimensional concept, but its properties can be easily shown in a 3D space by projecting a 4D unit hyper-sphere to a 3D sphere using a hyper-plane which is orthogonal to the real axis of the quaternion and a stereographic projection. Note that due to the fact that the shown space is a hyper-plane, the beginning of the coordinate system is quaternion number $[1, 0]$. This representation is shown also in figure 3.3. According to equations (3.27) through (3.30), point i left-multiplicated by j goes to $-k$, point k left-multiplicated by j goes to i. Point 1, which lives in the origin of the coordinate system, becomes j. 30

3.3 An example of a right multiplication of quaternion $[0, -j]$ in a 3D projection and left multiplication of quaternion $[0, j]$ (already shown in figure 3.2). According to equations (3.27) through (3.30), point i right-multiplicated by $-j$ goes to $-k$, point k right-multiplicated by $-j$ goes to i. Point 1, which lives in the origin of coordinate system, becomes $-j$. The total effect of the multiplication of the left quaternion and the right conjugate quaternion is canceled along the j axis, where the real value of the projection center was entangled. Conversely, the effect in jk is doubled. 31
3.4 Sample SLAM problem modeled as a factor graph. Variable nodes are marked as circles and factors as squares. Gray color marks hidden variables. 38
3.5 Figure is showing one edge of Graph SLAM. The $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are poses, which are hidden variables. There is an observation of pose $\boldsymbol{x}_j$ from pose $\boldsymbol{x}_i$ with value $\boldsymbol{z}_{ij}$ and information matrix $\boldsymbol{\Omega}_{ij}$. The observation allows to project an image $\boldsymbol{x}'_j$ of $\boldsymbol{x}_j$ respect to location $\boldsymbol{x}_i$ and observation $(\boldsymbol{z}_{ij}, \boldsymbol{\Omega}_{ij})$. Gray ellipse is confusion ellipse that visualizes information matrix. Error that this edge contributes is distance from image $\boldsymbol{x}'_j$ to location $\boldsymbol{x}_j$. 39
3.6 Graph SLAM problem before optimization and after optimization. Dataset is provided in [24] and software solution was used [2]. 40
3.7 The distribution of translation error for axes 'X' and 'Y' in residuals in Graph SLAM problem shown in figure 3.6. Both of histograms presents a bell-shaped distribution. Those are symmetric, but suffer from excess kurtosis. The top distribution has kurtosis 3.76 and bottom 12.77. The high kurtosis is caused by large number of outliers and very long and fat tails. To summarize, those distributions are not Gaussian. 43
3.8 Robust kernel function and weight in function of residual value. 45
3.9 Simulated calibration scene for synthetic equivalent of system shown in figure 4.6. Two cameras Red Green Blue Depth (RGBD) cameras are marked. High Dynamic Range (HDR) map in background imitates real-world illumination. 49
3.10 The plot of Root Mean Square Error (RMSE) after a finished iteration of the Iterative Closest Point (ICP) on a synthetic data-set for multiple initial condition. Various colors represent different initial errors. The violet plot, which fails to optimize, represents the largest initial error. The first method (top) uses a symbolically computed Jacobian using SymPy [84]. The second one (bottom) uses an analytical Jacobian on a Lie algebra with equation (3.103). Each initial error was introduced by increased rotation. Thus, both methods converge only for the limited initial error. For more significant errors, several iterations are too few. Otherwise, the optimization fails to converge at all. Both methods are sufficient, but the perturbation model Jacobian provides better stability and robustness. . . 52

3.11 The plot of error after the finished iteration of the ICP on synthetic dataset. The convergence of Levenberg-Marquardt ICP is compared for different Jacobians. All these optimizers find a parameter which lives in $\mathfrak{se}(3)$ space. The red plot shows a RMSE during the optimization using an automatic Jacobian [2]. The green and blue show the RMSE during optimization using an analytical Jacobian. The first one is a symbolical differentiation of Rodrigues formula (3.18) using SymPy [84]. The second one is a Jacobian on a Lie algebra which is given in equation (3.103). The symbolic and analytical Jacobian on a Lie algebra gives similar results, while analytical on a Lie algebra is more computationally efficient. 53
3.12 The line segment with (yellow) and vectors that creates Plücker coordinate. Moment vector $\boldsymbol{m}$ is marked in red and directional vector $\boldsymbol{l}$ is marked in green. Note that $\boldsymbol{m}$ length is two times $\boldsymbol{l}$ length. That due to fact of yellow line to be 2 units away from begin of coordinate system. . 55
3.13 The line segment in a 3D space is marked in yellow. There are two vectors that create the Plücker coordinates: the moment is marked in red and the directional vector is marked in green. The projection plane spans on the center of the projection sphere (marked in green) and the observed line (yellow). This plane contains a great circle (marked in blue) of the projection sphere. A normal vector of the great circle is marked in blue. . 56
3.14 Great circle projection equation (3.115) for various values of inclination and azimuth. 57
3.15 The line segment in a 3D space is marked in yellow. There are two vectors that create the Plücker coordinate: the moment is marked in red and the directional vector is marked in green. The projection plane spans on the center of the projection (marked in green) and the observed line (yellow). This plane contains an image of the line (marked in blue). A normal vector to projection plane is marked in blue. 61

4.1 Mechanical structure of the high-volume mobile 3d scanner. Actuated revolution joint u_1, actuated prismatic joints: u_2, u_3. Revolution joints with hidden configuration: c_1,c_2,c_3,c_4. 64
4.2 Processing of fiducial markers. 68
4.3 Detected fiducials before and after automatic calibration. 69
4.4 Factor graph modeling the calibrated system. U - controls (configuration of actuated joints). x_0, x_1, x_2, x_4 - poses of calibrated camera. $l1, l2$ - observed centroids. $\theta = \{c_1, c_2, c_3, c_4\}$ - calibrated parameters (configuration of assembled joints). 70
4.5 One of the two scanning heads. There is a visible guiding system with a timing belt in the back. In the front, there is an RGBD sensor with an Red Green Blue (RGB) camera and LiDAR lens. 70

4.6 The complete calibrated system. The calibration field is a combination of a large number of Augmented Reality (AR) markers. Mechanically, the system consists of two guided rails with a timing belt linear transport for two scanning heads marked in red. The arm on the bottom is rotated by a Direct Current (DC)-servo motor (curved red arrow). The system can be assembled in multiple ways to achieve the best coverage of the scanned volume. 71
4.7 Converging cost with iterations of the Levenberg-Marquadt algorithm for synthetic and real dataset. The cost is total distance. 14710 observations were used in the real dataset. 1074 observations were used in the synthetic one. 72
4.8 3D Computer Aided Design (CAD) model of assembly. 1- hexagonal pyramid; 2-Livox LiDAR; 3- face plate; 4- one of six mirrors. 73
4.9 Reshaped field of view. A plot of vertical situation. 74
4.10 Reshaped field of view, plot of horizontal situation. 74
4.11 The impact of the mirror angle on the changing vertical field of view. . . 75
4.12 Geomerical interpretation of equation (4.9). $\mathbf{b}^d$ is a unit vector representing the direction of the incident beam. $\mathbf{r}^d$ is a unit vector representing the direction of the reflected beam. $\mathbf{V}^{pl}$ represent a mirror's normal vector. The dot product of $\mathbf{b}^d \cdot \mathbf{V}^{pl}$ is the length of the projection of $\mathbf{b}^d$ on $\mathbf{V}^{pl}$. Finally, the sum of vectors $-2(\mathbf{b}^d \cdot \mathbf{V}^{pl})\mathbf{V}^{pl}$ and $-\mathbf{b}^d$ results in $\mathbf{r}^d$ vector. 77
4.13 The experimental prototype mounted onto a precise rotating table during the data acquisition procedure. 'A' marks the axis for changing the rotation angle. 78
4.15 Result of calibration algorithm for single measurement station without ground truth data (simplest scenario). 81
4.16 Result of the calibration algorithm for multiple measurement stations with histogram of errors. 82
4.17 Synchronization registers and signals in synchronization solution for the Livox Lidar. 84
4.18 Rotating mirror assembly. 1-Livox Mid-40, 2- incremental encoder, 3- mirror support, 4- Motor housing, 6- top plate, 5-bottom plate, 7- pillars. 85
4.19 Quantitative evaluation of the calibration procedure. 86
4.20 Comparison of the calibration results of the redundant model which optimizes $\left[\begin{smallmatrix} \boldsymbol{R}^c & \boldsymbol{t}^c \\ \mathbf{0}^{1\times 3} & 1 \end{smallmatrix}\right]$, $[a, b, c, d]$ and extrinsic poses, and the reduced model which optimizes $[a, b, c, d]$ and extrinsic poses. The redundant model is drawn in green and the reduced one is drawn in red. The reduced model converges to a sub-optimal solution. 87
4.21 Drawing of the lightweight 3D scanning system. 90
4.22 Pulse Per Second (PPS) (yellow) and TX (violet) signal show in oscillogram PPS signal is a short pulse with the frequency of 1 Hertz, TX is EIA RS-232C transmission with 9600 baud-rate containing 'GPRMC' National Marine Electronics Association (NMEA) message with simulated timestamp and position. Velodyne VLP 16 laser scanner is synchronizing its internal hardware timer to that signal. 91

4.23 Block diagram of the system. Ethernet is marked in green, with different physical layers marked. The user has direct access to Velodyne VLP-16 TCP/UDP ports. Velodyne VLP-16 consumes synchronization messages from STM32F1 with pair of NMEA messages and PPS signal, that latches transmitted timestamp. Physically, NMEA message is sent on one TX line of EIA RS-232C with 9600 baud rate PPS signal is TTL square signal with frequency of 1 Hertz with a duty smaller than one. Oscillogram of such signal is shown in figure 4.22. The encoder output (quadrature pair with reference) is connected to STM32F1's timer-counter channels and feedback input of the motor controller. In this application a single encoder is used as measurement and control. Controller Area Network (CAN) network of two nodes is marked in gray. Note that CAN network is used only inside the device, so the device's firmware implements only some segments of CiA 301 standard. 92
4.24 Detailed view of not calibrated field measured by the system. Bright colors correspond to the saturated points. Green points are the points that were measured by the left side of the laser scanner. Red points are points that were measured by the right side of the laser scanner (so-called side "A" and "B"). The drawn coordinate system that is visible in the middle is a coordinate system found by Principal Component Analysis (PCA)/Singular Value Decomposition (SVD) of the saturated points, and the quadrant of XY sub-system are marked. 93
4.25 Left - photo of the calibration pattern, right top - projection of detected patterns using initial (CAD) calibration, right bottom - projection of detected patterns using optimized calibration. 95
4.26 3D pointcloud before and after calibration. Red and green color represent sides of laser scanner. 96
4.27 A mobile mapping system, mounted on the top turntable. 1-spherical camera, 2-top laser scanner, 3-center laser scanner,4-side laser scanner, 4-IMU, 6-side laser scanner (transport protection was not removed in the photo), 7-turntable with crank. 98
4.28 Steps of extrinsics calibration of a laser scanner. Top left - input data, visible only TOP laser scanner. Top right - the optimized trajectory of calibrated system. Bottom left - trajectory optimized, but non calibrated. There is a large visible discrepancy between the scans marked in green and red. Bottom right - the calibrated system. Control pads (retroreflector, marked with bright colors) are getting closer, which is a qualitative improvement. 99
4.29 Calibration results. First row - camera image, middle row - CAD based calibration, bottom row - calibrated data. Green - center laser, red - side laser. Note that retroreflector images become more aligned along the process of calibration. 101
4.30 Calibration results. Left column - uncalibrated data, right column - calibrated data. The histograms build on top of evaluation of calibrated measurements are consistent in sense of variance and mean. 102

4.31 Calibrated system in left. There is a visible structure from extrusions that are easily detected by a laser scanner in the back. There is a point cloud captured by a laser scanner on the right with detected lines marked in colors. The gizmo manipulator shows the current state of calibration. . 104
4.32 Input image. Please note stiching error changing with depth. The effect is similar to that shown in figure 1.6. Manually selected points are marked with stars. The next projection curve was fitted and plotted with lines. . 104
4.33 Pre-calibrated image with projected lines feature which were detected in the point-cloud. 105
4.34 Post-calibrated image with projected lines feature that were detected in point-cloud. 105
4.35 Calibrated extrinsic of spherical camera used for applying texture to point-cloud. 106
4.38 Color point-cloud, different stitching methods. 111
4.39 Projection of calibration line features on equirectangular image. 112

5.1 Photo and drawing of the shaft mapping probe. 115
5.2 Effect of trajectory constraint in automatic calibration. Both maps are registered, but differs in trajectory and recovered calibration. Left trajectory is jagged, right is smooth. To successfully perform an automatic calibration of such system, trajectory constrains are needed. Those constraints ensure that parameters regarding calibration will be optimized instead of individual trajectory nodes. 116
5.3 Vertical intersection of shaft level, black - high-precision stationary scan, red - Livox Tele-15, green - Velodcyne VLP-32C, purple - rotated Velodyne VLP-16. 117
5.4 Horizontal intersection of shaft level, black - high-precision stationary scan, red - Livox Tele-15, green - Velodyne VLP-32C, violet - rotated Velodyne VLP-16. 117
5.5 Accuracy assessment of calibrated robotic mobile mapping system against high-precision stationary scan. 119
5.6 Robotic mapping system used for the scenario. 121
5.7 Screenshot from the operator's console during scenario. 121
5.8 3D map obtained with robotic mobile mapping system in Enrich 2021 trial with stop-scan. 122
5.9 Used factor graph. Factors from f_1 to f_6 are odometry factors. Factors from f_{11} to f_{16} are observation factors. Factors from f_{31} to f_{36} Inertial Measurement Unit (IMU) prior factors. Factors from f_{20} is a loop closure. Note that observation factor that connects pose $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ not exists, due to failed Normal Distributions Transform (NDT) matching. The variables $\boldsymbol{u}_1$ to $\boldsymbol{u}_6$ are robot odometry readings. The variables $\boldsymbol{s}_1$ to $\boldsymbol{s}_6$ are scans taken at near the corresponding poses. 123
5.10 Tool that builds and optimize factor graph for a SLAM problem. The trajectory is not yet optimised. 123
5.11 3D map obtained with robotic mobile mapping system in Enrich2021 trial with mobile mapping. 124

5.12 A robotic platform used. 1- Clearpath Jackal mobile robot, 2- Velodyne VLP-16, 3- Livox Lidar with planar reflectors. 126
5.13 The result of mapping with the architectural blueprint. The trajectory is marked. 127
5.14 Comparison of input data. The yellow points are obstacles in map, the green are obstacles in incoming data from sensor, the white are ground points in data from sensor. Grey dots are positions of particles. 129
5.15 Comparison of localization performance depending on input data. . . . 130

List of Tables

4.1 Comparison of cumulative error on four calibration datasets. For every row, one dataset was used to obtain calibration and rest was used to assess accuracy. 67
4.2 The optimized parameters (planar reflectors' coefficients). 80
4.3 The optimized parameters before and after calibration. 96
4.4 Comparison of median error to ground truth before and after calibration. 103

5.1 Absolute Trajectory Error (ATE) for multiple localization method. 130

List of Abbreviations

6-DOF	**S**ix **D**egree **o**f **F**reedom
AHRS	**A**ttitude and **H**eading **R**eference **S**ystems
AI	**A**rtifitial **I**ntelligence
APD	**A**valanche **P**hoto**d**etectors
API	**A**pplication **P**rogramming **I**nterface
AR	**A**ugmented **R**eality
ARM	**A**dvanced **R**ISC **M**achine
ATE	**A**bsolute **T**rajectory **E**rror
BA	**B**undle **A**djustment
BLDC	**B**rushless **D**irect-**C**urrent Motor
BSDF	**B**idirectional **S**cattering **D**istribution **F**unction
CAD	**C**omputer **A**ided **D**esign
CAN	**C**ontroller **A**rea **N**etwork
CNC	**C**omputerized **N**umerical **C**ontrol
CPU	**C**entral **P**rocessing **U**nit
CV	**C**omputer **V**ision
DC	**D**irect **C**urrent
DEM	**D**igital **E**levation **M**odel
DGPS	**D**ifferential **G**lobal **P**ositioning **S**ystem
DLT	**D**irect **L**inear **T**ransform
DSLR	**D**igital **S**ingle **L**ens **R**eflex Camera
DTM	**D**igital **T**errain **M**odel
EKF	**E**xtended **K**alman **F**ilter
FDM	**F**used **D**eposition **M**odelling
FLANN	**F**ast **L**ibrary for **A**pproximate **N**earest **N**eighbors
FOG	**F**iber-**O**ptic **G**yroscope
FOV	**F**ield **O**f **V**iew
GLONASS	**Glo**bal′naya **Na**vigatsionnaya **S**putnikovaya **S**istema
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
GSM	**G**lobal **S**ystem for **M**obile Communications
HD	**H**igh **D**efinition
HDR	**H**igh **D**ynamic **R**ange
ICP	**I**terative **C**losest **P**oint
IMU	**I**nertial **M**easurement **U**nit
INS	**I**nertial **N**avigation **S**ystem
IRLS	**I**teratively **R**eweighted **L**east **S**quares
IRU	**I**nertial **R**eference **U**nit

ISO	The **I**nternational **O**rganization for **S**tandardization
IT	**I**nformation **T**echnology
KF	**K**alman **F**ilter
LiDAR	**Li**ght **D**etection **a**nd **R**anging
LIO	**L**iDAR **I**nterial **O**dometry
LM	**L**evenberg-**M**arquardt
LRG	**L**aser **R**ing **G**yros
MEMS	**M**icro**e**lectro**m**echanical **S**ystem
MMS	**M**obile **M**apping **S**ystem
NDS	**N**avigation **D**ata **S**tandard
NDT	**N**ormal **D**istributions **T**ransform
NDVI	**N**ormalized **D**ifference **V**egetation **I**ndex
NMEA	**N**ational **M**arine **E**lectronics **A**ssociation
NMT	**N**etwork **M**anagmen**t**
OEM	**O**riginal **E**quipment **M**anufacturer
PCA	**P**rincipal **C**omponent **A**nalysis
PCL	**P**oint **C**loud **L**ibrary
PDF	**P**robability **D**istribution **F**unction
PF	**P**article **F**ilter
PID	**P**roportional **I**ntegral **D**erivative
PLA	**P**oly**l**actic **A**cid
PLD	**P**ulse **L**aser **D**iodes
PLY	**P**o**ly**gon File Format
PMMA	**P**oly **M**ethyl **M**eth**a**crylate
PPS	**P**ulse **P**er **S**econd
PTP	**P**recision **T**ime **P**rotocol
PWM	**P**ulse **W**idth **M**odulation
QR	**Q**uick **R**esponse
RADAR	**RA**dio **D**etection **A**nd **R**anging
RFID	**R**adio **F**requency **ID**entification
RGB	**R**ed **G**reen **B**lue
RGBD	**R**ed **G**reen **B**lue **D**epth
RMS	**R**oot **M**ean **S**quare
RMSE	**R**oot **M**ean **S**quare **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
ROI	**R**egion **O**f **I**nterest
RPE	**R**elative **P**ose **E**rror
RPM	**R**evolutions **P**er **M**inute
RSSI	**R**eceived **S**ignal **S**trength **I**ndicator
RTK	**R**eal-**T**ime **K**inematics Positioning
SDK	**S**oftware **D**evelopment **K**it
SDO	**S**ervice **D**ata **O**bject
SI	**I**nternational **S**ystem of Units
SLAM	**S**imultaneous **L**ocalisation **a**nd **M**apping
SLERP	**S**pherical **L**inear Int**erp**olation
SONAR	**SO**und **N**avigation **A**nd **R**anging

SVD	**S**ingular **V**alue **D**ecomposition
TLS	**T**errestial **L**aser **S**canning
TOF	**T**ime **o**f **F**light
UART	**U**niversal **A**synchronous **R**eceiver **T**ransmitter
UAV	**U**nmanned **A**erial **V**ehicle
UDP	**U**ser **D**atagram **P**rotocol
UGV	**U**nmanned **G**round **V**ehicle
UKF	**U**nscented **K**alman **F**ilter
USAR	**U**rban **S**earch **a**nd **R**escue
VO	**V**isual **O**dometry
XGA	**E**x**t**ended **G**raphics **A**rray

Chapter 1

Introduction

This dissertation concerns automatic calibration of multi-sensor robotic mobile mapping systems. Said calibration reduces the amount of expert knowledge required in the process and increases the autonomy of mobile robots performing tasks in unknown environments. The primary motivation for this dissertation is to provide a methodology as complementary set of methods which enables building multi-sensor systems for ground mobile robots. These methods focus mainly on fundamental problems such as online data synchronization, automatic calibration, and multi-view data registration; thus, the online capabilities extend the existing mobile mapping systems. In order to verify the proposed methodology, several ground robotic mapping systems were built and tested in realistic conditions, including a nuclear power plant inspection during the European Robotics Hackathon: ENRICH [123]. This provided the opportunity to explore the workings of a mapping system capable of processing data on-board a robot and using it for additional tasks such as semi-autonomous navigation. It was, therefore, possible to perform the mission remotely, which ensured safety of the personnel. The primary practical benefit of the methodology is enabling the use of mobile mapping data for ground robot navigation, which is what differentiates the operational capability of this robotic mapping system from other state-of-the-art mobile mapping systems. The direct positive impact on the mobile robotics domain has been verified as mobile robot localization shows that the proposed methodology can be incorporated in various mobile robotic applications. This research fills the gap in recent mobile robotics studies by showing the importance of data synchronization and automatic calibration, which yield highly accurate mobile mapping results. A formal definition of calibration given by The International Organization for Standardization (ISO) in the Internal Vocabulary of Basic and General Terms in metrology is: "Calibration (V-IM-1993) - set of operations that establish, under specified conditions, the relationship between values of quantities indicated by a measuring instrument or measuring system, or values represented by a material measure or a reference material, and the corresponding values realized by standards [98]". In Computer Vision (CV) there are multiple types of calibration [134]:

- intrinsic calibration,
- extrinsic calibration,
- hand-eye calibration,
- photometric calibration.

Intrinsic calibration is the model parameter of the sensor. This type of calibration allows to remove undesirable effects on the captured image introduced by imperfections in a lens. This collection of parameters describes a chosen model of lens distortion (e.g. radial and tangential). Extrinsic calibration is related to relative poses between sensors expressed in the local coordinate system of the measurement instrument. That calibration parameter depends on the geometrical configuration of the system. It is required by any sensors - IMUs, gyroscopes, Global Navigation Satellite System (GNSS) antennas, cameras, laser scanners, range finders. It can be optimized during automatic or manual procedures or obtained from the documentation. Hand-eye calibration is a problem of determining a transformation between the end-effector, camera, and base of the robot, and the world. It is an essential problem in tasks involving automated robotic manipulation [121]. Photometric calibration is a function which converts the sensor's measurements (e.g. pixel value) to photometric quantities (e.g. International System of Units (SI) light units). For a camera, these would be: color gain correction, vignetting gain correction, or hot pixel correction.

To conclude, intrinsic and extrinsic calibration often require [112][133] the use of an existing calibration field, patterns, or other objects with known geometrical properties. An alternative approach [31] assumes no prior knowledge of calibration field or patterns, as any data association is not available. Unfortunately, there is no general approach for the automatic calibration of multi-sensor systems, which is the primary concern of this thesis. Thus, the main focus is to discuss the sensors, then the methodology, and finally the experimental validation. Additional assessments in real applications demonstrate the added value of the approach.

1.1 Thesis outline

The thesis is organized into six chapters. Chapter Introduction is introductory and presents addressed sensors and environment representations. The problem and theses are formulated in chapter Problem formulation. The methodology for the automation of the multi-sensor system calibration process is given in chapter Methodology. Above-mentioned chapter contains detailed discussion of state of the art method and tools and presents new introduced tools for automated calibration. The experimental validation of proposed methodology is discussed in chapter Experimental validation, in which several multi sensor prototypes are evaluated. Furthermore, important robotic applications are described in chapter Robotic applications, which provides evidence for the added value of the research. Chapter Conclusions concludes the dissertation and addresses future directions of the research of robotic mobile mapping systems.

1.2 Sensors in ground robotic mapping systems

1.2.1 LiDAR

LiDAR is a method of measuring distance which implements Time of Flight (TOF) technology. LiDAR is widely applied in road survey and underground mapping, as well as in Digital Terrain Model (DTM) and Digital Elevation Model (DEM) generation used for, among others, land survey. Multiple LiDARs solutions are used in robotics and robotic mapping systems. In order to achieve a desired FOV, multiple measuring devices may be used at once. The shape of FOV can be used to differentiate the design of LiDAR sensors. One of such are 2D lasers, which are already in use in the 2D robot navigation. The first mode of their operation is based on a rotating 1D laser scanner. Such simple solution is used typically in consumer-grade household devices. The second design method is based on a rotating planar reflector.

Another application for the devices is in safety equipment, in case of which strict quality control is of utmost importance. They accurately detect any intrusion into the hazardous zone and reliably trigger safety functions. LiDAR was developed more than 20 years ago and it is considered a mature technology. A popular approach is a rotating laser scanner where the head spins at a rate 100-200 Revolutions Per Minute (RPM). The rotating head consists of a pair of Pulse Laser Diodes (PLD) and Avalanche Photodetectors (APD). These are assembled on the focal plane of two lenses. A beam produced by PLD travels through the lens, and having hit the target, it returns through the second lens and is focused on APD. This approach gives an omnidirectional field of view with limited azimuth. Conversely, another approach called retina-like is much simpler to manufacture. It consists of a pair of Risley prisms that are placed next to each other, assembled with Brushless Direct-Current Motor (BLDC). The prisms rotate at slightly different rates. With a constantly changing phase shift, a beam is pointed in a direction depending on the angular offset of two Risley prisms. The comparison of the two methods is described in [77].

1.2.2 Cameras

There are multiple vision sensor and camera types that are used in the robotic mapping system. The latter can be differentiated based on a number of criteria. The first division focuses on the spectrum of detected light:

- visible light,
- near-infrared,
- infrared (thermografic),
- multimodal.

Visible light cameras detect wavelengths which match the abilities of the human's eye (400-700 nanometers). Cameras detecting the near-infrared (700 nm - 1100 nm) part of the electromagnetic spectrum help capture image in poor-light conditions. Near-infrared cameras find application in agriculture and vegetation studies to detect live green vegetation. Output from these cameras can be provided as so-called Normalized Difference Vegetation Index (NDVI), which expresses the ratio of red light to near-infrared light. Healthy plants reflect large amounts of light in the near-infrared spectrum, which is not used for photosynthesis, and absorb the visible part of the spectrum used in photosynthesis. It is so due to a mechanism in plants which allows them to absorb as large amount of useful energy as possible, thus limiting the increase in their temperature. Inanimate objects, such as soil, snow, or dead plants, absorb the same amount of light in the near-infrared and visible spectrum. NDVI is widely used in satellite imagery and in specialized multi-modal cameras used in agriculture, e.g. in drone mapping [16]. Infrared cameras - or thermographic cameras - detect wavelengths longer than it is possible for near-infrared sensors, that is 750 nm - 14 000 nm. A black body produces the electromagnetic radiation of that wavelength at a given temperature. Therefore, thermographic cameras are often tuned to produce an output in temperature units of a black body. This methodology finds application in military, civil engineering, firefighting [119], medicine [18], or inspection tasks. Cameras can be differentiated also by the shape of their field of view:

- perspective (rectilinear) model camera,
- fisheye model camera,
- omnidirectional model cameras.

An ideal rectilinear camera performs a projection of a point in the 3D world to a 2D pixel plane. Camera parameters are optical lengths f_x, f_y. Rectilinear projection maintains linearity - the line observed in the scene is projected as the line in the image. If the lens is not ideal, a projected line is a curve. If a camera is ideally rectilinear (no distortion) and optical length is common for f_x and f_y , the distance from the projection center is a function of angle θ of incident light ray [10]:

$$r = f \tan(\theta) \tag{1.1}$$

This equation shows that the rectilinear model has a field of view limited to 180 degrees. What is more, the infinite size sensor is needed to observe the ray that enters the lens at an angle close to 180 degrees. A model of an actual camera is not a perfect perspective projection. For these cameras, the image requires undistortion. Pixels from the raw image need to be shifted to obtain an undistorted image that follows the rectilinear model. The distortion models and the methods to identify the model parameters are discussed in [133][109]. Next, fish-eye is an ultra-wide lens producing a highly distorted or hemispherical image. Its field of view is, in some cases, larger than 180 degrees [10]. The most popular fish-eye cameras follow two projection models. The first is equidistant:

$$r = f\theta \tag{1.2}$$

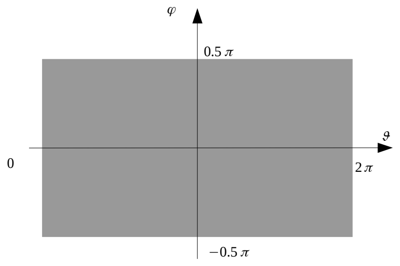
The second is equisolid:

$$r = 2f \sin(\theta/2) \tag{1.3}$$

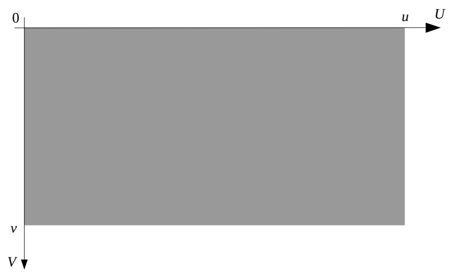
The first model is beneficial in celestial body observation - it allows for an efficient computation of angles, which can be further preserved identically in all images [10]. An omnidirectional camera (such as a spherical camera) has a field of view larger than fish-eye cameras. The spherical camera produces an equirectangular image of a surrounding scene. Limitations of those types of the sensor are discussed in chapter 1.2.2. Several publications discuss usage of those type of sensors in various different domains, such as heritage prevention [52] [118], or robotic SLAM [117]. An extensive discussion on the limitation of a spherical image in surveying and photogrammetry was discussed in [74].

Spherical cameras

The spherical camera model uses an equirectangular (or cylindrical) projection of the observed scene. This model of the camera projects a terrain point to its spherical coordinates. A row and a column in a spherical image (panorama) correspond to:



(A) Equirectangular coordinate system of spherical image.



(B) Image coordinate system of spherical image.

FIGURE 1.1: Coordinate system used in equirectangular image.

- ϑ as longitude,
- φ as latitude.

Typically, the spherical image is 2π in width and π in height. The angular coordinate system used when such image is processed is shown in the figure 1.1a. The pixel coordinate system is shown in 1.1b.

$$\begin{bmatrix} \vartheta \\ \varphi \end{bmatrix} = \begin{bmatrix} \frac{\pi u - 0.5U}{U} \\ \frac{\pi v - 0.5V}{V} \end{bmatrix} \tag{1.4}$$

where: u, v - coordinates of the pixel; U, V - number of columns and rows in the panoramic image.

The point is at a distance of R from the optical center of the spherical camera. Its coordinates in $\mathbb{R}^3$ are given in equation (1.5) [74]:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R \sin\varphi \sin\vartheta \\ R \sin\varphi \cos\vartheta \\ R \cos\varphi \\ 1 \end{bmatrix} \tag{1.5}$$

Implementation of spherical camera

While the ideal equirectangular camera does not exist, there are three major techniques for designing a spherical capture system:

- a single camera mounted on a gimbaled tripod or motorized platform, which may be either simply a Digital Single Lens Reflex Camera (DSLR) or a specialized fully calibrated system,
- multi-lens (dioptric) cameras system with two or more fish-eye lenses, e.g. cameras such as Flir Ladybug 5/5+,

- mirror systems (catadioptric)[107].

There is a technical limitation that introduces a discrepancy between a model and a real system, as it is impossible to bring the optical centers of the individual camera to the optical center of the multicamera system. That limitation has several interesting consequences. A multi-lens system contains multiple perspective cameras with wide lenses. An example of such camera is Flir Ladybug 5/5+. It consists of six individual wide FOV cameras, which are intrinsically and extrinsically calibrated. Rectification is performed by a proprietary algorithm which is exposed by the provided Software Development Kit (SDK). Manufacturer provides a local coordinate system for every camera. Composing the panoramic image from multiple cameras is a straightforward process. With known cameras' extrinsic and intrinsic parameters, the point in 3D space can be easily transformed to 2D sensor coordinates. Every point in a projection sphere with a fixed radius of R has pair of spherical coordinates ϑ, φ. The equation (1.5) allows to transform those coordinates to $\mathbb{R}^3$. Equation (1.6) transforms point ϑ, φ on the sphere with R radius to local coordinates system of certain perspective camera.

$$\begin{bmatrix} x_{nc} \\ y_{nc} \\ z_{nc} \\ 1 \end{bmatrix} = \mathbf{T}_{ne} \begin{bmatrix} R\cos\vartheta\sin\varphi \\ R\sin\varphi\sin\vartheta \\ R\cos\varphi \\ 1 \end{bmatrix} \tag{1.6}$$

where: $\mathbf{T}_{ne} \in SE(3)$ - extrinsics calibration of n-th camera. $[x_{nc}, y_{nc}, z_{nc}] \in \mathbb{R}^3$ - projected point to local coordinate system of n-th camera.
Local coordinates of pixel in perspective camera are expressed in equations (1.7) and (1.8).

$$u = f_{nx}\frac{x_{nc}}{z_{nc}} + c_{nx} \tag{1.7}$$

$$v = f_{ny}\frac{y_{nc}}{z_{nc}} + c_{ny} \tag{1.8}$$

where:
u, v - the coordinates of the projected pixel in rectified image of n-th camera,
f_{nx}, f_{ny} - optical length in x and y direction for n-th camera,
c_{nx}, c_{ny} - principal point location for n-th camera.
It is evident that chosen value of R affects the result [74]. This effect can be observed in figures 1.4 and 1.5 in relation with an ideal case shown in figure 1.3. Those three figures are panoramas projected from a simulated six-camera rig shown in figure 1.2. Note that simulated cameras are ideal pin-hole cameras with known extrinsics and intrinsics. The method of simulation, the used software, its benefits and limitations are described in chapter 3.3.6. Used scene with lighting, texture and meshes was obtained commercially [14]. Figures 1.4 and 1.5 differ - especially on borders, and are both far from an ideal equirectangular camera output shown in 1.3. Moreover, figure 1.6 demonstrates the local comparison of the stitching errors. That parallax error is most visible on the stitching line, but exists at every point of the panoramic image. The error disappears when the projected sphere has a radius equal to the distance to the object. In figure 1.7 geometrical reasoning for the parallax error is shown. The value of the parallax error in function of distance and radius of projection sphere is given with equation (1.9).

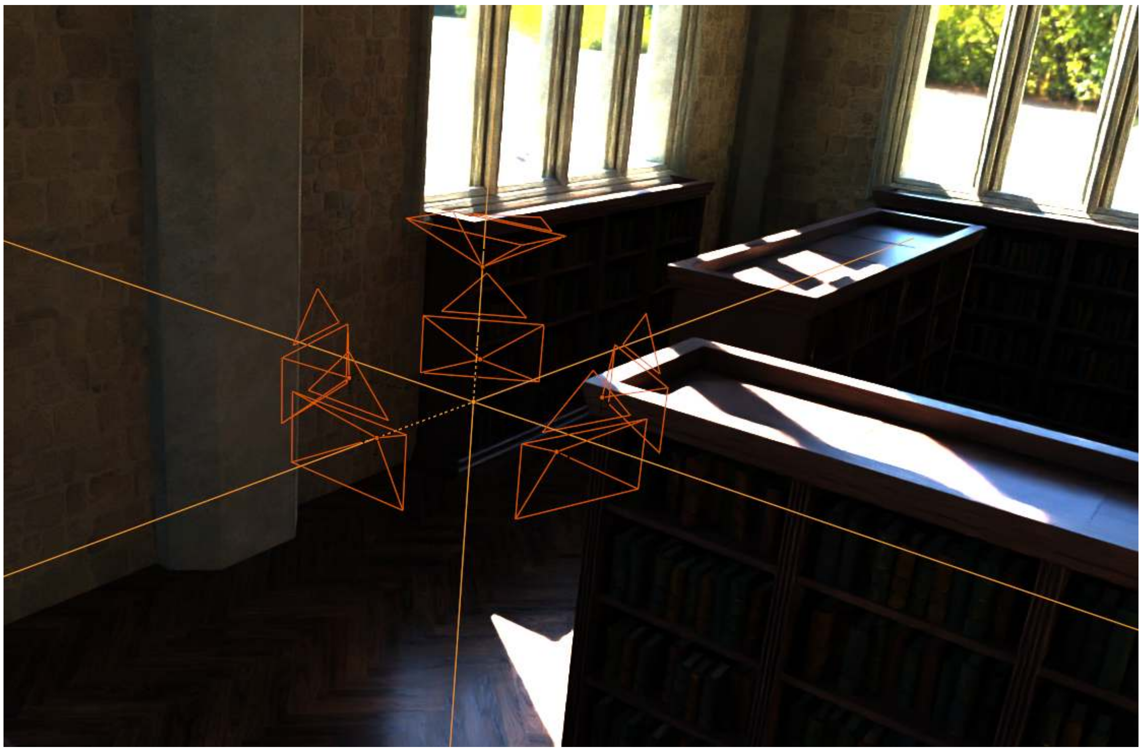


FIGURE 1.2: Simulated spherical camera rig with t = 0.05m.

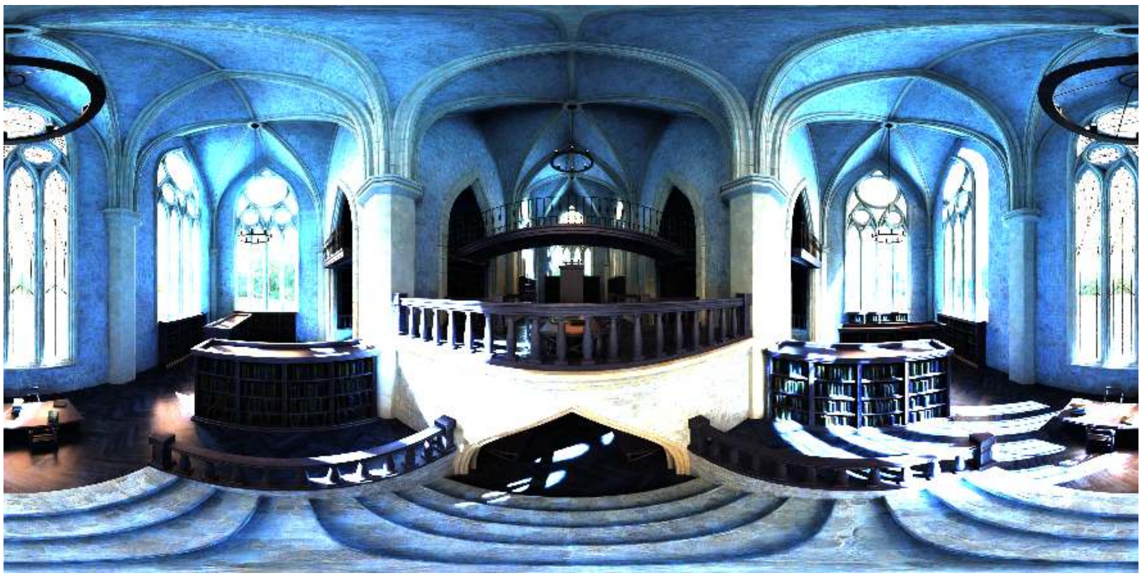


FIGURE 1.3: Panorama rendered with an ideal equirectangular camera.

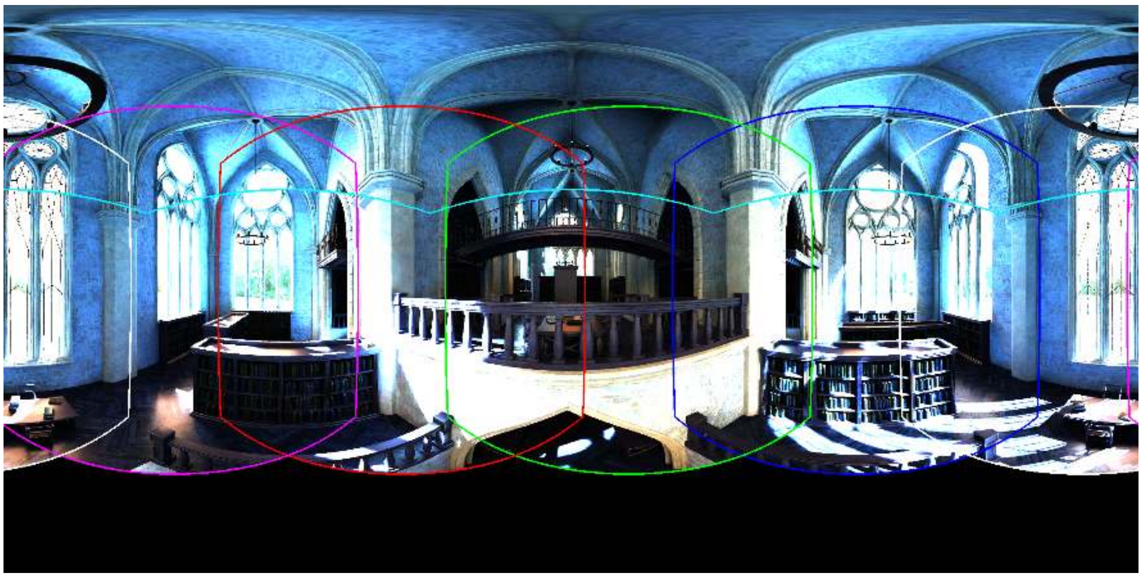


FIGURE 1.4: Panorama rendered with six camera rig from figure 1.2, projected on sphere with radius 2 meters. Projected field of view boundaries are marked in color.

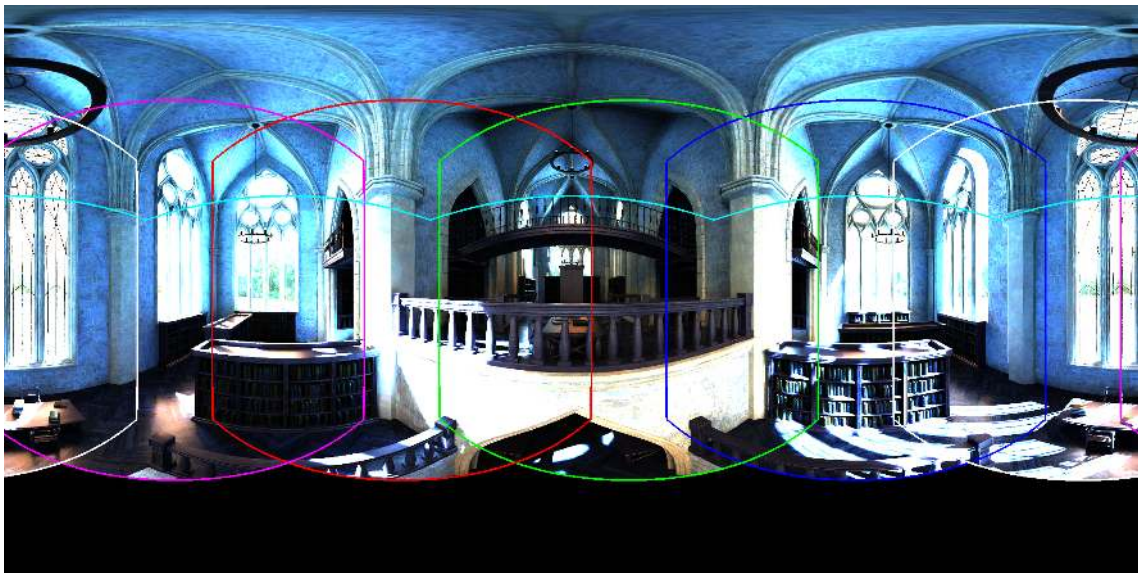


FIGURE 1.5: Panorama rendered with six camera rig from figure 1.2, projected on sphere with radius 10 meters. Projected field of view boundaries are marked in color.

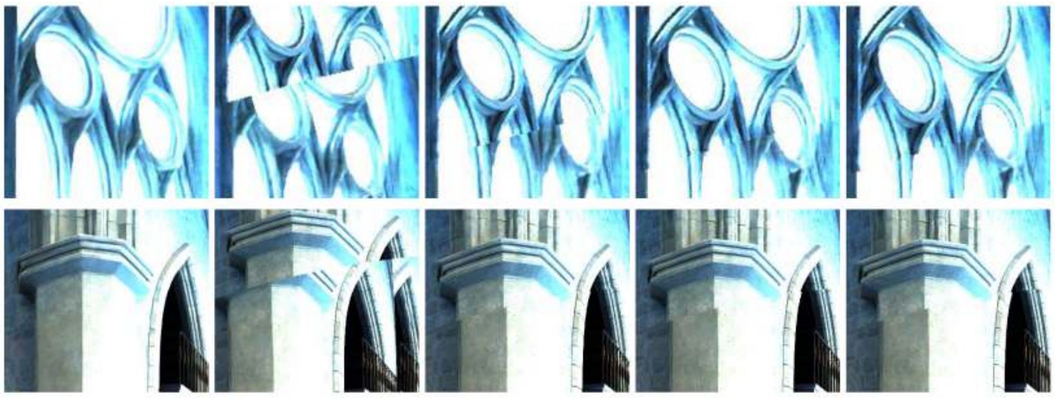


FIGURE 1.6: Local comparison of stitching errors. Left column is ground truth obtained with the ideal equirectangular camera simulation. Next columns are projections for sphere size: 1, 5, 10, 100 meters. Note that optimal radius is different for those two patches.

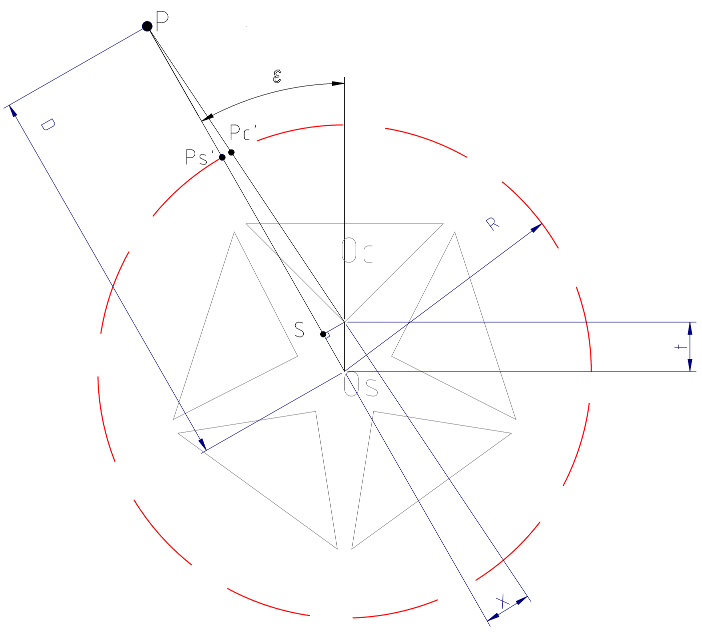


FIGURE 1.7: Parallax error 'X'. Point P is projected to a point P'_s in an idealized spherical model. In real camera point P is projected to P'_c. That is caused by a fact, that projection center for spherical image is Os and for perspective camera is Oc. Those are t apart.

The equation is obtained from the similarity of triangles P, P'_s, P'_c and P, S, O_c visible in figure 1.7 [67]. In plot 1.8 it is visible that the parallax error becomes zero when the radius of the projection sphere is equal to a distance to the object. The error is smaller for larger distances. Note that the error becomes large and changes rapidly for objects that are close to the camera.

$$X = \frac{(D - R)t\sin(\varepsilon)}{D} \tag{1.9}$$

where:
D - the distance to object, R - the radius of sphere, t - the distance of the optical center of the camera and the spherical model, ε - the angular location in image.
Note that in presented simulated data wide lens cameras produced rectified images,

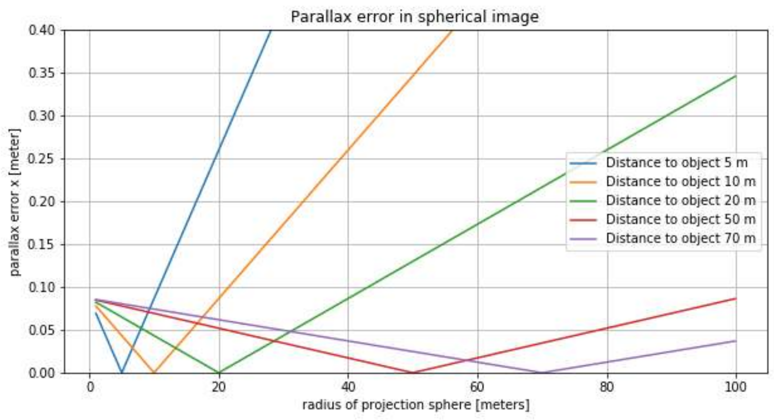


FIGURE 1.8: The value of parallax error in meters for given camera $\varepsilon = 60°$ and $t = 0.1m$.

which is not the case in the real world. The wide FOV lenses suffers from significant distortions. The producer of Ladybug camera claims that their proprietary algorithm can produce an undistorted image with the accuracy of 2 pixels in the border of the picture. That error further contributes to the spherical projection error, especially on the borders.

1.2.3 IMU

IMU is one of the major measurement instrument in mobile robotics and mobile mapping systems. It consists of multiple sensors:

- accelerometers,
- gyroscopes.

and optionally:

- magnetometer for Attitude and Heading Reference Systems (AHRS),
- barometer (for height estimation),

- thermometer (for temperature compensation).

The range of application of IMU is extremely wide. Medical and biomedical applications are especially interesting. Due to high measurement frequency and small sizes of Microelectromechanical System (MEMS) and minimal power consumption, the wearable IMU system can provide unique data. The system is used to analyze the gait of patients [90] or performance of sportsmen [110]. The next application worth mentioning is for the entertainment industry, where a small wearable system can replace the motion capture setup in the studio. This allows to lessen the cost of outdoor motion capture sessions [65]. Multiple commercial wearable solutions are available on the market for various applications. IMU system is essential in Unmanned Aerial Vehicle (UAV) application [40][39] for state estimation. Unmanned Ground Vehicle (UGV) navigation utilizes IMU measurement in state estimation as well [87]. The specialized inertial system provides measurements in Electronic Stability control in road vehicles. Specialized low-bias inertial sensor systems are essential for Inertial Reference Unit (IRU) in aviation. Those sensors allow performing inertial navigation of an aircraft using a self-contained system that is entirely independent of radio navigation and GNSS. Small MEMS sensors are essential in mobile devices [100] for virtual (and augmented) reality applications [85].

A part of IMU essential for state estimation of the orientation of a body in space is the gyroscope. There are multiple technologies used for these sensors: Mechanical, where a rotor is mounted on a low-friction gimbal; the direction of the spinning rotor is preserved due to the law of the conservation of angular momentum. This kind of sensor suffers from friction that introduces biases. In addition, the construction of the gimbal can introduce a phenomenon called gimbal lock. Laser Ring Gyros (LRG) operates on basis of the Sagnac effect [95]. It consists of a single laser that produces two light beams pointing in opposite directions. These two beams are directed into the same point by two mirrors and meet at a common point where they interfere with each other. One light beam travels clockwise, the second one in counter-clockwise direction. When the system is still, two beams travel precisely at the same speed. Thus, the traverse of both sides of the ring takes identical time, and the interference image shows no phase shift. When the system is under rotation, one beam must travel a greater distance than the other. The speed of light is constant, so two beams "arrive" at a common point at a different moment. That shows a phase shift in interference image [95]. The Fiber-Optic Gyroscope (FOG) system also utilizes Sagnac's effect. The construction of a light pathway is different - a FOG IMU consists of multiple fiber optic cable loops. The main difference is that a phase shift for the same rotation velocity would be multiple times larger in FOG than in LRG [95]. MEMS introduces a lithographically constructed mechanism of vibrating masses. It is a self-containing integrated circuit [95]. The choice of the technology depends on the task. For example, LRG IMUs are found in applications, such as aviation and aeronautics navigation. Those are cheaper and simpler than FOGs. On the other hand, FOG IMUs are more complex and can be scaled to application. The bias stability for FOG can be extremely small (even 0.00008 deg/h). Due to excellent stability, those sensors find application in precise navigation (submarine, spacecraft), metrology, seismology, structural sensing and calibration equipment [71]. MEMS sensors provide a small, durable solution in the form of an integrated circuit. Those sensors, depending on their type, have worse bias stability than FOG or LRG (1 - 1000 deg/h). MEMS technology has, however, certain advantages: low production cost and insignificant energy consumption.

MEMS gyroscopes utilize the Coriolis effect and vibrating mass. A proof mass is induced to vibrate. Under rotation, the mass is displaced by the Coriolis effect. This displacement changes capacitance, which is measured by the electronics of the gyroscope. The exact implementation of MEMS differs by implemented structure. MEMS technology is being developed and gradually becomes a more preferable solution for applications that used to utilize optic IMUs [34][47][29][50]. Bias instability causes IMU (especially MEMS) to lose alignment, which manifests itself in every application. An IMU in a satellite needs to be aligned periodically using celestial bodies [122]. Another approach is AHRS with a magnetometer. The calibrated magnetometer detects direction to the Earth's magnetic North Pole. It is an absolute measurement of direction in 3D space. Accelerometers used in IMU utilizes MEMS technology. It measures a displacement of a proof mass. Using a sophisticated application of Extended Kalman Filter (EKF), a more reliable system can be obtained [131].

1.2.4 Odometry

Odometry is a measurement of a change of position and orientation over time. The word "odometry" is combination of Greek *odos* (meaning "route") and *metron* (meaning "measurement"). Localizing a robotic platform in an environment requires the use of dead reckoning, that is a technique in navigation that enables the estimation of a location of a moving object from the knowledge that consists: previous localization, speed, heading (course), and elapsed time. This simple method suffers from an unbounded, cumulative error which needs to be minimized with other techniques, such as using visual aids, celestial body navigation, radar stations or GNSS. Dead reckoning in ground robotic systems is based on encoder sensors attached to the platform's motors, which are used to measure achieved velocity of robot's wheels and, after applying simple forward kinematics computations, provide an estimated location of a robotic platform over time [82]. Terms such as "Visual Odometry (VO)" and "LiDAR odometry" appear in literature. Those methods provide updates of relative position using data from other domains. They also suffer from cumulative, unbounded error. Odometry measurement can be fused with other sensors' data (e.g. IMU), creating an inertial localization system using EKF [87] or Unscented Kalman Filter (UKF)[38], algorithms which allow finding an optimal solution to robot localization. Odometry is also functional in SLAM as an initial guess and the relative poses which express constraints [73].

1.2.5 GNSS

GNSS is a constellation of artificial satellites which provides full coverage for the Earth. It consists of:

- European Galileo,
- United States NAVSTAR Global Positioning System (GPS),
- Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS),
- China's BeiDou Navigation Satellite System.

GNSS systems are critical for providing accurate location service and accurate time sources. In a nutshell, a GNSS receiver obtains a signal with a timestamp from several multiple geostationary satellites. The receiver needs to have a clear line of sight to satellites. Having at least four messages from different satellites consisting of timestamps, knowing the speed of light, time of arrival, the system can triangulate its locations. The problem is complex because the receiver needs to have its time source synchronized with the satelites. What is more, the location of every satellite needs to be corrected. Currently, modern small Original Equipment Manufacturer (OEM) modules can provide the user with 2 meters accuracy, but this number degrades to 15 - 20 meters under unfavorable conditions [58]. The GNSS system can provide its user with centimeter-level accuracy, but multiple techniques are required for the correction of so-called common errors in GNSS. These errors include [68]:

- selective availability (limited clock correction for civilian use),
- satellite clock error, relativistic effects on satellite clock,
- receiver clock errors,
- ionosphere transport delay that varies on Sun's activity, weather, and time of day,
- trophosphere errors that vary on humidity and weather,
- Sagnac effect on Earth rotation,
- multipath error, where receiver suffers from signal reflection from obstacles.

The abovementioned errors are some of the sources that degrade the GNSS localization. Fortunately, mobile mapping and surveyor applications can rely on a solutions that provide centimeter accuracy, described in next section.

RTK and DGPS

Real-Time Kinematics Positioning (RTK) and Differential Global Positioning System (DGPS) are methods that use a precisely localized GNSS receiver called a base station, which is localized using a different method (e.g. forward intersection). The second receiver (called rover) is used to measure a surveyed position. Base station aggregates measurements and distributes corrections over the radio, or Global System for Mobile Communications (GSM) link to the rover. That way, some of the common errors, e.g., atmospheric transport delays, can be eliminated. While a converged RTK fix provides centimeters accuracy, DGPS solution provides decimeter accuracy [75], which leads to RTK being a more advanced approach that expands the concept of DGPS. In RTK base and rover the receivers analyze phase shift of carrier frequency to improve accuracy even further [41].

GNSS+INS

GNSS+Inertial Navigation System (INS) expands GNSS localization with the benefits of IMU. It provides the user with a full Six Degree of Freedom (6-DOF) pose and can provide dead reckoning localization during GNSS outages, e.g. temporarily GNSS denied environments. It is more robust against GNSS spoofing and jamming. It is a standard for GNSS solutions to use EKF, but Recurrent Neural Network (RNN) is currently being researched [30].

1.3 Map representation

Map representation is a crucial design decision in autonomous mobile robots. Recent advances in sensors and on-board processing allow to incorporate different approaches, such as dense and sparse representations, that are elaborated in next sections.

1.3.1 Dense representation

Occupancy grid map

Occupancy grid map is a popular representation of environment used in SLAM and localization approaches such as Particle Filter (PF). It is a 2D dense representation used to quickly model the posterior probability of map state given robot trajectory and measurements. Occupancy grid map is an image base container with metadata that scales the occupancy grid map to represent the environment metrically. Typically, bright pixels show free spaces, and dark pixels show obstacles. This representation is discretized into cells, each of which acts as a single independent random variable [124].

Elevation map

2.5D map is a simplified grid-based model of environment that can incorporate 3D measurements from depth camera or 3D laser scanners. Every cell contains the height of the potential obstacle. It is also called the elevation model. Like occupancy grid map, it can serve multiple purposes: localization, mapping or motion planning. It can be used both in mobile ground robots and aerial robots [91].

1.3.2 Sparse representation

Topological map

A topological map is a simplified representation that lacks scale but shows interesting entities and relations. Such representation can be a layer in a more sophisticated representation (e.g., line-string in High Definition Map). Topological representation of the environment is memory efficient and enables effective path planning. For example, some authors claim that topological representation built from voxel occupancy grid map can provide path planning an order of magnitude faster compared to dense representation [15]. Topological maps can encode knowledge regarding the structure of more complex environments, e.g. clustering available space to separate rooms, floors, or a particular abstract region of interest, such as portals. Portals are places where a robot can safely transit from one region to another. Topological map in robotic system is favorable due to the following:

- it provides the knowledge that cannot be abstracted in other representation,

- in the past, lightweight representation was desired due to limited computational resources [72],
- topological model with artificial neural network can provide robot localization [32].

Semantic map

Semantic map can use previously described representations with an additional layer of information. It can be organized as semantic net [94]. It is worth mentioning that using the semantic information [4] extracted from the 3D laser data [93] was an interesting research topic in mobile robotics. In [92] a semantic map for a mobile robot was described as a map that contains assignments of mapped features to entities of known classes. In [54] a model of a scene is implemented as a knowledge base that can be considered also as a semantic net. Some researchers were focused on automatic feature extraction for semantic conceptualization [23][42][128].

1.3.3 High Definition map

A High-Definition map represents an environment tailored for ego-vehicle localization, motion planning, and autonomous vehicle safety [21]. Ego-vehicle is an automated vehicle that is carrying sensors and is primary interest in scenario. It is designed to support vehicle autonomy (from level 3 to level 5 of SAEJ 3016). This representation is a concept that is implemented in a number of ways, depending on system architecture and used solutions. It is a model that consists of multiple layers of abstraction. Multiple formats exist and they depend on the provider. A popular format that is currently used by several manufacturers is Navigation Data Standard (NDS) provided by NDS Association. It is a wide, closed-source navigation database, developed jointly by the users and map providers. In addition, NDS Association provides the users with standard tools and certification solutions for new products [5]. Another example is Lanelet2, an open-source solution provided as C++ library [99]. Lanelet2 uses following primitives for capturing a road geometry:

- line-string,
- polygons,
- lanes,
- area (such as parking spots),
- regulatory elements (traffic lights).

Note that many map providers deliver their solutions offering different features and data layers, more sophisticated than Lanelet2. An interesting aspect of several keys is the concept of self-healing and knowledge sharing between agents, that uses High Definition (HD) Map [59]. This solution allows to modify existing maps with real-time observations. The concept is sufficient in situations such as rerouting due to accidents or heavy traffic. HD Map building involves designing robotic mapping system and mobile mapping system. It requires sensor calibration, solution for data acquisition and registration, SLAM modeling and solving, Artifitial Intelligence (AI)-based and manual labeling to provide raw content to be supplied as HD Map [125].

Chapter 2

Problem formulation

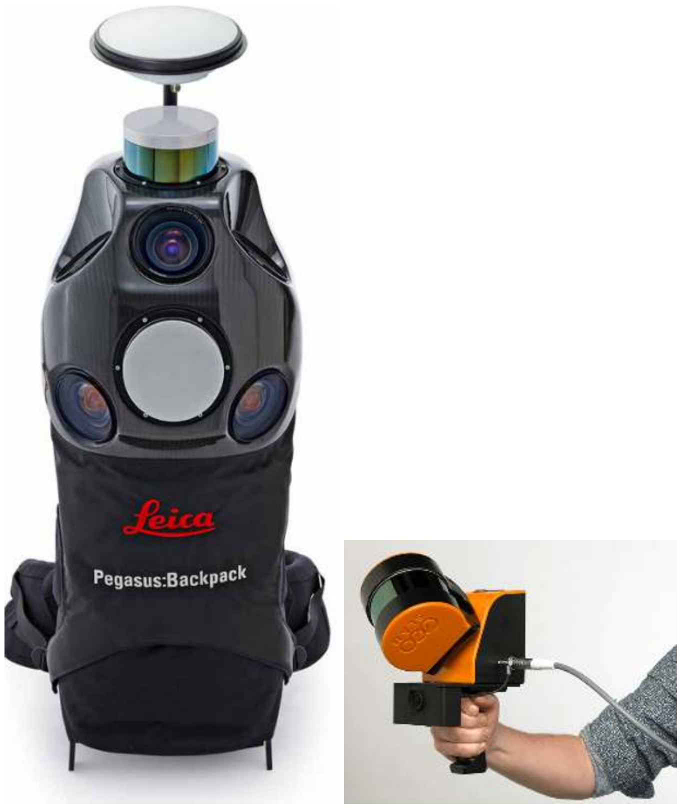
Robotics perception is essential to update the model of a robotics agent belief about its environment. Updating the model of the environment can be done in various ways, depending on the application. A single sensor or multi-sensory system can be calibrated in advance, or the calibration parameters can be considered unknown variables in later optimization, such as SLAM or Bundle Adjustment (BA).

Robotized mapping systems need to meet requirements different from those that are set for mobile mapping systems. They are the following:

- a robotic systems need to be robust to withstand accelerations and impacts from robotic platform locomotion solution,
- they need to be teleoperated, semi-autonomous, or fully autonomous,
- they need to operate with a small bandwidth and large communications delays,
- the same sensor for mapping and navigation should be used to limit power consumption and mass,
- direct cooperation with the operator may be limited or not possible at all (e.g. Urban Search and Rescue (USAR), exploration rovers, surveying),
- systems often need to perform pre-planned missions (e.g. unmanned aerial mapping),
- in some scenarios (e.g. USAR) a map delivery is needed on-demand during the mission.

A MMS is a device or a whole system designed to maximize output map quality with fewer constraints. The MMS has to be lightweight and ergonomic, and it needs to provide a smooth experience to the operator. These systems are often a combination of multiple sensors that measure the internal state of the system (e.g. IMU), and precept external environment in a passive (e.g., monocular cameras, stereo cameras, spherical cameras) or an active way (structure light, LiDAR, SOund Navigation And Ranging (SONAR) and Radio Detection And Ranging (RADAR)). MMSs are designed to work in both GNSS allowed and denied environments. The sensor suite is typically synchronized with the onboard clock and provides data storage for the measurement. Specifically designed and optimized algorithms similar to SLAM or BA are used to post-process data acquired by the MMS.

An example of a commercially available MMS is shown in the figure 2.1. A key



(A) Leica Pegasus MMS. (B) Zeb Horizon MMS.

FIGURE 2.1: Examples of commercially available MMSs.

feature of all MMSs is the calibration of all intrinsic and extrinsic parameters. There is no common methodology for doing so which can be formulated as an optimization problem. The main thesis of the dissertation addresses this key aspect and it is stated as follows: **Automation of the calibration process of the mobile mapping allows to obtain more accurate results from the mapping system, while reducing the expert knowledge required in the calibration process and increasing the autonomy needed by applying those systems in the field using mobile robots.** Four more precise theses have been formulated to support the main thesis:

- automation of the calibration process reduces the need for expert knowledge required for accurate measurement of intrinsic and extrinsic calibration parameters,
- the new method for reshaping the field of view of modern Solid State LiDARs enables customizing the robotic mobile mapping systems for various applications,
- the chosen rotation matrix parametrization enables robust optimization of the calibration parameters,
- automatic calibration enables long term autonomous mobile robot inspection of the unknown environment by reducing mechanical issues related to the robot's exploitation.

Chapter 3

Methodology

In this chapter a number of key aspects such as trajectory representation, observations equations are presented as a core of proposed methodology. Those aspects are an important part of system design and automatic calibration presented later in this thesis. In this chapter some well-established method such as Levenberg-Marquardt (LM) and factor graph are presented based on literature. Also there are addresses details that are dependent on domain and applications, and, what is more, can be approached in multiple ways. This chapter presents methodology that is implemented later for real-world challenges that were presented in chapters 4 and 5.

3.1 Time synchronization

A complex robotic mapping system contains multiple sensors that can produce data at different rates. Ideally, every measurement would be combined with an accurate timestamp provided by the hardware. The problem of accurate synchronization of trajectories is addressed in many applications [43][56].

Hardware time synchronization

There are multiple solutions for time synchronization in a mobile mapping system. One of them is PPS. This signal traditionally produced by GNSS receiver can provide hardware timestamp to a number of devices [130][79]. Chapters 4.3 and 4.4 discusses the detailed implementations.

A vast number of other existing solutions utilize technologies from Information Technology (IT). An example of one is Precision Time Protocol (PTP). This is a standard which enables synchronizing devices in a local computer network. Some hardware is PTP-enabled [79]. PTP is a standard for time synchronization in other industries, e.g. financial transactions, communication or power grid equipment [132]. Many simpler sensors used in robotic mapping systems are based on custom solutions for precise time synchronization. For example, IMU outputs high-frequency data-stream over its serial port. One of the synchronization strategies exposed to the user is the so-called trigger indicator. For example, in a status word being a part of a serial transmission, there is a bit which shows if IMU's firmware detected a rising edge on one of the digital inputs. With such a solution, the user can easily find the offset of the IMU internal timer and other parts of the user's system [89]. The importance of the hardware time synchronization is one of the main theses of this dissertation.

3.2 Trajectory

The trajectory of the robotic system is a set of robot poses which represent the robot's movements. Most SLAM systems utilize a trajectory built from $SE(3)$ or $SE(2)$ members. However, such trajectory can be more informative, which is the case with other systems, e.g. the LiDAR Interial Odometry (LIO) algorithm [113]. In this approach, the trajectory is not coming from the odometry of a robot but from the IMU itself. The pose is integrated, which results in an initial guess for the trajectory. Both the accelerometer and the gyroscope measurements contain bias. Estimation of this bias, along with the $SE(3)$ pose, composes the trajectory node. Therefore, SLAM front end not only performs scan matching, but also estimates the current bias of IMU. This approach also effectively integrates rotation velocity into the rotation matrix and acceleration into the linear velocity. Wheel odometry, VO, or IMU can be represented in many ways. In the case of factor graph SLAM, odometry is a factor function which utilizes the last pose, the next pose, and the displacement reported by odometry. An excellent example of building such a representation is available in [49]. Alongside other factors which utilize sensors measurements and loop closure detection, such odometry provides a structure for factor graph optimization. The pose graph SLAM approach is similar. Odometry is introduced to the pose graph SLAM problem in the form of an equation which reports how much the displacement measured by the odometry differs from the one estimated by the subsequent trajectory nodes. The so-called relative pose constraint for pose graph SLAM is discussed in section 3.2.2. Solving the SLAM problem results in the most probable optimized trajectory given all information that SLAM utilizes. Trajectory is crucial for mobile mapping systems since the map is reconstructed based on it. For this reason, the accuracy of the trajectory has a direct impact on the final map. As it is shown, calibration is another crucial factor in the process because it also contributes to map generation.

3.2.1 Rotation representation

Groups

A group is a term used to refer to several concepts that are used in robotics or in state estimation. For a set to be called a group, it needs to have a binary operator, also called the operation, which consumes two elements and yields a third one which satisfies the following axioms:

- closure,
- associativity,
- identity (or neutral),
- and inverse element.

The first axiom means that the operator cannot yield an element that does not belong to the group. A good example here is a finite set of integer numbers from -10 to 10: $\mathbb{Z}_1 = \{-10\ldots, 0, \ldots 10\}$. This set cannot be called a group under addition operation. It can return an integer number that is more than 10 or less than -10, which is not in $\mathbb{Z}_1$.

The second axiom means that the condition (3.1) is satisfied for all a, b, c in the group under operator $\circ$.

$$(a \circ b) \circ c = a \circ (b \circ c) \tag{3.1}$$

It is worth mentioning that the commutativity of the operator $\circ$ is not needed, and often non-existent.

The third axiom means that a set needs to contain some unique element that does not change the result of the operator, e.g. identity matrix in matrix groups under multiplication, or zero in groups under addition. In other words: a group needs to contain an element I that for every a in some group under operator $\circ$, $(a \circ I) = a$.

Finally, the fourth axiom means that there is always an element which provides inverse action. In matrix groups, an inverse matrix satisfies this condition under multiplication. Note that groups of matrices need to be invertible in the first place. In groups under addition that is a negative element. In other words: for every element a in the group under $\circ$ needs to exist some element a_i which satisfies: $a \circ a_i = I$ and $a_i \circ a = I$ where I is the identity element.

Groups can be finite or infinite (continuous). They can consist of numbers, vectors, or matrices. Some good examples used in robotics [115] are:

- unit complex numbers under multiplication S^1,
- orthogonal three by three matrices under multiplication $SO(3)$,
- normalized quaternion numbers under multiplication S^3,
- n-D vector planes under addition.

Lie Groups

Some of the groups are Lie groups. A Lie group needs to have a differentiable manifold. In other words, all available elements in the group should create some smooth hypersurface. An example of such is a circle created by all complex numbers with magnitude that equals to one - the S^1 group. Such circle is a unit circle, and every point on it is locally differentiable. A smooth surface enables creating a chart, also called a tangent space, at any given point. A tangent space and a manifold have only one point in common. Therefore, only one tangent space can exist for a given point in the manifold. The tangent space which is attached to the Identity element is called a Lie algebra. A Lie algebra is a vector space. This greatly simplifies algebraic operations like differentiation (both analytic and numeric). In other words, considering a Lie group $SO(3)$, every point on the manifold is a three by three rotation matrix. A Lie algebra is a vector space where every point can be mapped back and forth the manifold.

Lie group theory provides one with a number of tools [115]:

- exponential map,
- logarithmic map,
- vee $(.)^\vee$,
- hat $(.)^\wedge$.

An exponential map allows for a precise conversion of a point that lies on a tangent space back on the manifold. A logarithmic map is an inverse operation to an exponential map. It allows for converting a point on a manifold (in the group) to a Lie algebra. Those two operations are the result of solving an ordinary differential equation for a manifold.

Let us look at group S^1. It is a two-dimensional group whose manifold lives in the form of a unit circle on a complex plane. A manifold constraint is:

$$z^*z = 1 \tag{3.2}$$

Only a complex number which satisfies the equation (3.2) belongs to the S^1 group. Its algebra $\mathfrak{s}^1$ is simply a tangent line going through the point $(1, i0)$ in a complex plane parallel to the imaginary axis. Every point θ on this line can be mapped to a circle:

$$e^{i\theta} \in S^1 \tag{3.3}$$

This exponential mapping, which is the well-known Euler formula [88], always gives a complex number with a module equal to one for every real θ. This property is extensible to all Lie groups and algebras. In the application in which a Lie algebra is used for working with a rotation of the rigid body, an exponential can be computed in a closed-form solution (e.g. Euler formula or Rodrigues formula [45]). In the case of S^1, it is obtained as follows:

$$e^{i\theta} = \sin\theta + i\cos\theta \tag{3.4}$$

An interesting practical fact pertaining to a Lie algebra properties is that algebra members can be applied to group members directly. The above holds for tiny local displacements: it is used for derivation and in the numerical Jacobian. A group element multiplied by a small algebra member effectively takes only the first two elements of the Taylor expansion given by equation (3.5).

$$e^{i\theta} = 1 + ix + \frac{(ix)^2}{2!} + \frac{(ix)^3}{3!} + \dots \tag{3.5}$$

This will be shown for the less trivial case and can be used effectively in some applications, e.g. numerical derivation.

Naturally, applying large displacements will give elements above the manifold, e.g. non-unit complex numbers in case of S^1. 'Hat' and 'vee' are commonly used mathematical notations [45][115][11]. 'Hat' is an isomorphism which maps a vector (τ) in real space to a Lie algebra.

$$\tau \in \mathbb{R}^1; (\tau)^\wedge \in \mathfrak{s}^1 \tag{3.6}$$

A trivial case of S^1 hat operator is:

$$(\theta)^\wedge = i\theta \tag{3.7}$$

The isomorphism 'vee' ($(.)^\vee$) is an opposite to 'hat'. It maps Lie algebra members into a real vector space. A particularly elegant physical interpretation of a Lie algebra is a velocity. The velocity vector has similar properties: the velocity vector is always tangent to the trajectory. For this reason, terms such as "twist" are sometimes used for a Lie algebra in the case of rigid body motion. A Lie algebra can be effectively used for manipulating every element in the Lie group. It is also used as a tool that enables easy manipulation of a group element X with operator $\oplus$ into another Y, where X and Y belong to the Lie group. Operator $\oplus$ is defined with equation (3.8). This operator converts applied vector τ into increment which lives in the Lie algebra.

$$Y = X \oplus \tau = X \circ e^{(\tau^{\wedge})} \tag{3.8}$$

There is also a left implementation of $\oplus$ - equation (3.9).

$$Y = \tau \oplus X = e^{(\tau^{\wedge})} \circ X \tag{3.9}$$

There is a linear relation of left $\oplus$ and right $\oplus$ operators. For the same starting point X on the manifold, there exist vectors τ_1 and τ_2 which lead to the same result on the manifold:

$$X \oplus \tau_1 = \tau_2 \oplus X \tag{3.10}$$

$$\tau_1 = Ad_x \tau_2 \tag{3.11}$$

Where Ad_x is called the adjoint of the group at X.

For a simple group like S^1 the adjoint is always unit, which is caused by commutativity of the operator multiplication in complex number domain. There are left and right operators $\ominus$ which for given two points in the manifold yield a displacement in the algebra. More details on this can be found in [115].

Special orthogonality group $SO(3)$

A special orthogonality group $SO(3)$ is a set of all available rotations. $SO(3)$ under multiplication is a Lie group. Group axioms are easily observable in the properties of a rotations matrix. There is an identity matrix of 3×3 which serves as a neutral element. The group consists of orthogonal matrices which are always invertible. The closure means that two rotations can be combined into one with multiplication. Rearranging the parentheses in an expression does not alter the result, which is shown in equation (3.12). Such property is a group axiom called associativity.

$$(\boldsymbol{X}_1\boldsymbol{X}_2)\boldsymbol{X}_3 = \boldsymbol{X}_1(\boldsymbol{X}_2\boldsymbol{X}_3) \tag{3.12}$$

where: $\boldsymbol{X}_1 \in SO(3)$,$\boldsymbol{X}_2 \in SO(3)$,$\boldsymbol{X}_3 \in SO(3)$.
Note that $SO(3)$ group elements, like every representation of rotation in 3D space, have no commutativity. $SO(3)$ is a Lie group. It means that it is differentiable and it creates a smooth manifold. The manifold does not have any singularities, which means that a slight change of its elements should not cause a great rotation. The manifold constraint ensures orthogonality. For every member of $SO(3)$ the following holds:

$$\mathbf{R}^{\intercal}\mathbf{R} = \boldsymbol{I} \tag{3.13}$$

Matrix $\mathbf{R}$ describes rotation in a 3D space; it is orthogonal and has a determinant equal to one. These properties need to be taken into consideration in the optimization process, where an optimal solution is searched for. Note that the dimension of the rotation matrix is 9, but it has only 3 degrees of freedom. It is difficult to operate with such a representation without tools like a Lie algebra. Changing any element in the matrix without coordination with its other elements leads to a matrix that is not orthogonal. The effect on a rotated body is shown in figure 3.1. Note that rotation is not rigid anymore if $det(R)$ is not equal to 1; thus, the object is deformed after applying this non-orthogonal matrix. Rotation matrices have the following properties which are a natural consequence of group axioms:

- multiplication of two elements in the group yields another element group,
- an inverse of a group element is another element of the group.

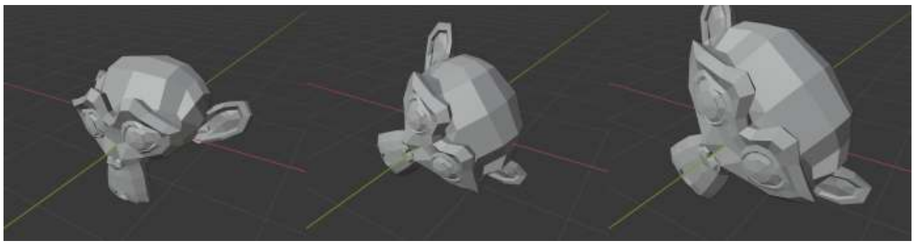


FIGURE 3.1: The example of applying non-valid ($\det(\boldsymbol{R}) = 1.5$) rotation matrix. Left: original object, middle: proper rotation, right: non-SO3 quasi-rotation. It is evident that the object is scaled.

Lie algebra $\mathfrak{so}(3)$

The $\mathfrak{so}(3)$ is a set of all three by three skew-symmetric matrices. The $\mathfrak{so}(3)$ algebra allows for performing multiplication of $SO(3)$ group member as summations on the tangential space $\mathfrak{so}(3)$. The important properties of every Lie algebra are exponential and logarithmic mapping. These two operations allow for converting an algebra to the corresponding group and vice versa. An exponential of an element of $\mathfrak{so}(3)$ gives $SO(3)$ member as in equation (3.14).

$$\exp\left(\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}^{\wedge}\right) = \exp\left(\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}_{\times}\right) = \exp\left(\begin{bmatrix}0 & -\omega_z & \omega_y\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix}\right) \in SO(3) \tag{3.14}$$

where vector $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^{\mathsf{T}}$ is a vector isomorphic to a member of $\mathfrak{so}(3)$. Operator 'hat' on vector $\boldsymbol{\omega}$ gives a $\mathfrak{so}(3)$ member : $(\boldsymbol{\omega})^{\wedge} \in \mathfrak{so}(3)$. The Lie algebra member $\boldsymbol{X} \in \mathfrak{so}(3)$ gives vector $\boldsymbol{\omega}$ with operator 'vee' : $(\boldsymbol{X})^{\vee} = \boldsymbol{\omega}$. The physical representation of vector $\boldsymbol{\omega}$ is a rotation velocity vector. The latter property of Lie algebras of a Lie group is, in this particular case, advantageous. It offers the possibility to perform optimization in 3D vector space of $\mathfrak{so}(3)$, where rotations are applied by addition by operators $\oplus$. Furthermore, taking the Lie algebra property into account, there is a possibility to compute a valid rotation matrix from optimized variables in every step. Note that optimizing parameters in $SO(3)$ space is non-trivial. It would require nine variables which need to coexist in a combination of constraints to ensure they keep a valid rotation, or rigid body motion. Every property of a Lie algebra discussed in the previous section holds here and can be taken advantage of. It is worth mentioning that $SO(3)$ does have a non-identity adjoint matrix-equation (3.15), so operators $\oplus$ and $\ominus$ behave differently when used as left and right. To keep both side of equations (3.16) equal, the Lie algebra members $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ need to be in relation (3.17).

$$\boldsymbol{Ad}_R = \mathbf{R} \tag{3.15}$$

$$\boldsymbol{R} \oplus \boldsymbol{\tau}_1 = \boldsymbol{\tau}_2 \oplus \boldsymbol{R} \tag{3.16}$$

$$\boldsymbol{\tau_1} = \boldsymbol{Ad}_R \boldsymbol{\tau_2} \tag{3.17}$$

where: $\boldsymbol{R} \in SE(3)$, $\boldsymbol{\tau_1} \in \mathfrak{so}(3)$ and $\boldsymbol{\tau_2} \in \mathfrak{so}(3)$.

Rodrigues' rotation formula for rotation matrix

Elements of $\mathfrak{so}(3)$ need to be exponentiated to yield $SO(3)$ Lie group elements (rotation matrix). The exponential skew-symmetric matrix has a closed-form solution which allows for computing exponential mapping without a Taylor expansion. The Rodrigues' rotation formula (3.18) is a way for closed-form exponential in $SO(3)$. It is equivalent to Euler's formula, which is closed-form exponential mapping in S^1 or $SO(2)$.

$$\boldsymbol{R} = e^{(\theta \boldsymbol{K})} = \boldsymbol{I} + (\sin\theta)\boldsymbol{K} + (1 - \cos\theta)\boldsymbol{K}^2 \tag{3.18}$$

where:

$$\boldsymbol{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \tag{3.19}$$

$\boldsymbol{R}$ - rotation matrix, $[k_x, k_y, k_z]$ - axis of rotation vector, θ - rotation angle.
Rodrigues' rotation formula formula can be inferred from known relation of $\mathfrak{so}(3)$ and $SO(3)$. An exponential function has the following Taylor expansion equation (3.20).

$$e^{\theta \boldsymbol{K}} = \boldsymbol{I} + \frac{\theta \boldsymbol{K}}{1!} + \frac{(\theta \boldsymbol{K})^2}{2!} + \frac{(\theta \boldsymbol{K})^3}{3!} + ... \tag{3.20}$$

Matrix $\boldsymbol{K}$ is a skew-symmetric matrix. This matrix has the following property, provided it is normalized $k_x^2 + k_y^2 + k_z^2 = 1$:

$$\boldsymbol{K}^3 = -\boldsymbol{K} \tag{3.21}$$

The property given with the equation (3.21) can be explained by investigating the substitution of individual vector elements into $\boldsymbol{K}^3$, which is given with equation (3.22).

$$\boldsymbol{K}^3 = \begin{bmatrix} 0 & k_z\left(k_x^2 + k_y^2 + k_z^2\right) & -k_y\left(k_x^2 + k_y^2 + k_z^2\right) \\ -k_z\left(k_x^2 + k_y^2 + k_z^2\right) & 0 & k_x\left(k_x^2 + k_y^2 + k_z^2\right) \\ k_y\left(k_x^2 + k_y^2 + k_z^2\right) & -k_x\left(k_x^2 + k_y^2 + k_z^2\right) & 0 \end{bmatrix} \tag{3.22}$$

where:
$[k_x, k_y, k_z] = \boldsymbol{k}$ - vector which builds skew-symmetric matrix $\boldsymbol{K}$. Next powers can be

computed using recursion and knowledge of $\boldsymbol{K}^3 = -\boldsymbol{K}$, e.g.:

$$\begin{aligned} \boldsymbol{K}^4 &= \boldsymbol{K}^3\boldsymbol{K} = -\boldsymbol{K}\boldsymbol{K} = -\boldsymbol{K}^2 \\ \boldsymbol{K}^5 &= \boldsymbol{K}^3\boldsymbol{K}^2 = -\boldsymbol{K}\boldsymbol{K}^2 = -\boldsymbol{K}^3 = \boldsymbol{K} \\ \boldsymbol{K}^6 &= \boldsymbol{K}^3\boldsymbol{K}^3 = (-\boldsymbol{K})(-\boldsymbol{K}) = \boldsymbol{K}^2 \end{aligned} \tag{3.23}$$

Based on the above, the multiplication exponential can be reformulated:

$$\begin{aligned} \exp(\theta\boldsymbol{K}) = \boldsymbol{I} + \frac{\theta\boldsymbol{K}}{1!} + \frac{(\theta\boldsymbol{K})^2}{2!} + \frac{(\theta\boldsymbol{K})^3}{3!} + \frac{(\theta\boldsymbol{K})^4}{4!} + \frac{(\theta\boldsymbol{K})^5}{5!} + ... = \\ \boldsymbol{I} + \theta\boldsymbol{K} + \frac{1}{2!}\theta^2\boldsymbol{K}^2 + \frac{1}{3!}\theta^3\boldsymbol{K}^3 + \frac{1}{4!}\theta^4\boldsymbol{K}^4 + \frac{1}{5!}\theta^5\boldsymbol{K}^5 + ... = \\ \boldsymbol{I} + \left(\theta\boldsymbol{K} + \frac{1}{3!}\theta^3\boldsymbol{K}^3 + \frac{1}{5!}\theta^5\boldsymbol{K}^5 + ...\right) + \left(\theta^2\boldsymbol{K}^2 + \frac{1}{4!}\theta^4\boldsymbol{K}^4 + \frac{1}{6!}\theta^6\boldsymbol{K}^6 + ...\right) = \\ \boldsymbol{I} + \left(\theta\boldsymbol{K} - \frac{1}{3!}\theta^3\boldsymbol{K} + \frac{1}{5!}\theta^5\boldsymbol{K} + ...\right) + \left(\theta^2\boldsymbol{K}^2 - \frac{1}{4!}\theta^4\boldsymbol{K}^2 + \frac{1}{6!}\theta^6\boldsymbol{K}^2 + ...\right) = \\ \boldsymbol{I} + \underbrace{\left(\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 + ...\right)}_{\sin(\theta)}\boldsymbol{K} + \underbrace{\left(\frac{1}{2!}\theta^2 - \frac{1}{4!}\theta^4 + \frac{1}{6!}\theta^6 + ...\right)}_{1-\cos(\theta)}\boldsymbol{K}^2 = \\ \boldsymbol{I} + \sin(\theta)\boldsymbol{K} + (1-\cos(\theta))\boldsymbol{K}^2 \end{aligned} \tag{3.24}$$

Equation (3.24) is a further derivation of a Taylor expansion (3.20). It is grouped in a way in which normalized skew symmetric is taken advantage of equations (3.23). These summation elements are grouped in a way which allows for substituting groups of summed elements with Taylor expansions of the following functions: $\sin(\theta)$ and $\cos(\theta)$. Finally, Rodrigues' rotation formula is obtained. Rodrigues' rotation formula exists also in a form which enables an efficient rotation of vector $\boldsymbol{v}$ to a new vector $\boldsymbol{v}_{rot}$ using a known axis of rotation $\boldsymbol{k}$ and angle of rotation:

$$\boldsymbol{v}_{rot} = \boldsymbol{v}\cos\theta + (\boldsymbol{k}\times\boldsymbol{v})\sin\theta + \boldsymbol{k}(\boldsymbol{k}\cdot\boldsymbol{v})(1-\cos\theta) \tag{3.25}$$

Rodrigues' rotation formula for vector rotation (3.25) has several applications in computer graphics and game engines.

Quaternions

Complex number multiplication allows for describing rotation in S^1. Multiplication of a real positive number by i changes its polar coordinates in the direction of the positive imaginary axis (counter-clockwise). It is an effective way to represent rotation in a 2D space, e.g. sinusoidal electric signal in the time domain or a process value in the control system, and it is an essential tool for describing processes in multiple engineering fields, such as electronics, controls, robotics. A quaternion is an extension of this concept. Complex numbers with the magnitude of 1 can represent all elements of $SO(2)$ space. Unit quaternions can represent all elements in $SO(3)$ space. The quaternion number is:

$$\mathbf{q} = q_0 + iq_1 + jq_2 + kq_3 \tag{3.26}$$

For all quaternion numbers the following equations are always satisfied:

$$i^2 = j^2 = k^2 = -1 \tag{3.27}$$

$$ij = k, ji = -k \tag{3.28}$$

$$jk = i, kj = -i \tag{3.29}$$

$$ki = j, ik = -j \tag{3.30}$$

Complex number multiplication is commutable. This property manifests itself in 2D representations of rotations other than unit complex numbers; therefore $SO(2)$ matrices can be multiplied in any given order. Conversely, to complex number multiplication, quaternion multiplication is not commutable. Multiple operations can be performed on quaternion numbers. Quaternions can be represented as an ordered pair:

$$\mathbf{q} = [s, \mathbf{v}] = [s, xi + yj + zk]; s, x, y, z \in \mathbb{R} \tag{3.31}$$

where :
s - the real part which lives in $\mathbb{R}$
$\mathbf{v}$ - the vector part which lives in ijk space.

A quaternion number is called a pure quaternion when its real part is equal to zero. A quaternion number is called real if its vector part, i.e. imaginary components, is equal to zero. A quaternion product is not commutable and always results in another quaternion, as seen in equation (3.32).

$$\mathbf{q}_a \cdot \mathbf{q}_b = [s_a, \mathbf{a}][s_b, \mathbf{b}] = (s_a + x_a i + y_a j + z_a k)(s_b + x_b i + y_b j + z_b k) \tag{3.32}$$

The above can be written using dot and cross products:

$$\mathbf{q}_a \cdot \mathbf{q}_b = [s_a s_b - \mathbf{v}_a^\intercal \mathbf{v}_b, s_a \mathbf{v}_a + s_b \mathbf{v}_b + \mathbf{v}_a \times \mathbf{v}_b]^\intercal \tag{3.33}$$

Quaternion multiplication manipulates an element in 3D imaginary space according to equations from (3.27) to (3.30). Imaginary components follow a circle in the rotation plane and change into the opposition in the rotation axis. It is shown in example of the left multiplication by a pure quaternion of $[0, j]$ in the figure 3.2. A sum of quaternions is similar to a complex number, where a sum of two quaternions is:

$$\mathbf{q}_a + \mathbf{q}_b = [s_a, \mathbf{v}_a] + [s_b, \mathbf{v}_b] = s_a + s_b + (x_a + x_b)i + (y_a + y_b)j + (z_a + z_b)k \tag{3.34}$$

The length of the quaterion is:

$$\|\mathbf{q}_a\| = \sqrt{s_a^2 + x_a^2 + y_a^2 + z_a^2} \tag{3.35}$$

The length of the quaternion has the following property:

$$\|\mathbf{q}_a\mathbf{q}_b\| = \|\mathbf{q}_a\| \, \|\mathbf{q}_b\| \tag{3.36}$$

A unit quaternion ($\|\mathbf{q_a}\| = 1$) stays unit when multiplied by another unit quaternion. A conjugate is a negation of the vector part of a quaternion, such as:

$$\mathbf{q}_a^* = [s_a, -\mathbf{v}_a] = (s_a, -x_a i, -y_a j, -z_a k) \tag{3.37}$$

Inverting a quaternion is:

$$\mathbf{q_a}^{-1} = \frac{\mathbf{q}_a^*}{\|\mathbf{q}_a\|} \tag{3.38}$$

To rotate point $\boldsymbol{p}_x$ into point $\boldsymbol{p}'_x$ using unit quaternion $\mathbf{q}_a$, first a pure quaternion representation of point $\boldsymbol{p}_x \in \mathbb{R}^3$ must be created:

$$\mathbf{s}_{px} = [0, \mathbf{p}_x]^\intercal \tag{3.39}$$

A rotation using a quaternion is a combination of two multiplications in a four-dimensional space.

$$\mathbf{s}_{p'x} = \mathbf{q}_a\mathbf{s}_{px}\mathbf{q}_a^* \tag{3.40}$$

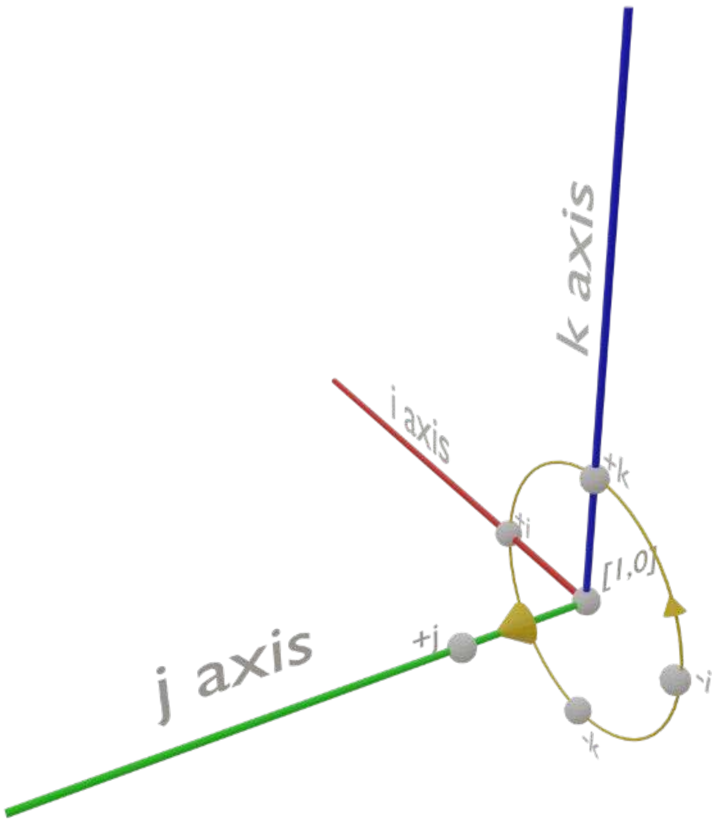


FIGURE 3.2: An example of a left multiplication of quaternion $[0, j]$ in a 3D projection. A quaternion is a four-dimensional concept, but its properties can be easily shown in a 3D space by projecting a 4D unit hyper-sphere to a 3D sphere using a hyper-plane which is orthogonal to the real axis of the quaternion and a stereographic projection. Note that due to the fact that the shown space is a hyper-plane, the beginning of the coordinate system is quaternion number $[1, 0]$. This representation is shown also in figure 3.3. According to equations (3.27) through (3.30), point i left-multiplicated by j goes to $-k$, point k left-multiplicated by j goes to i. Point 1, which lives in the origin of the coordinate system, becomes j.

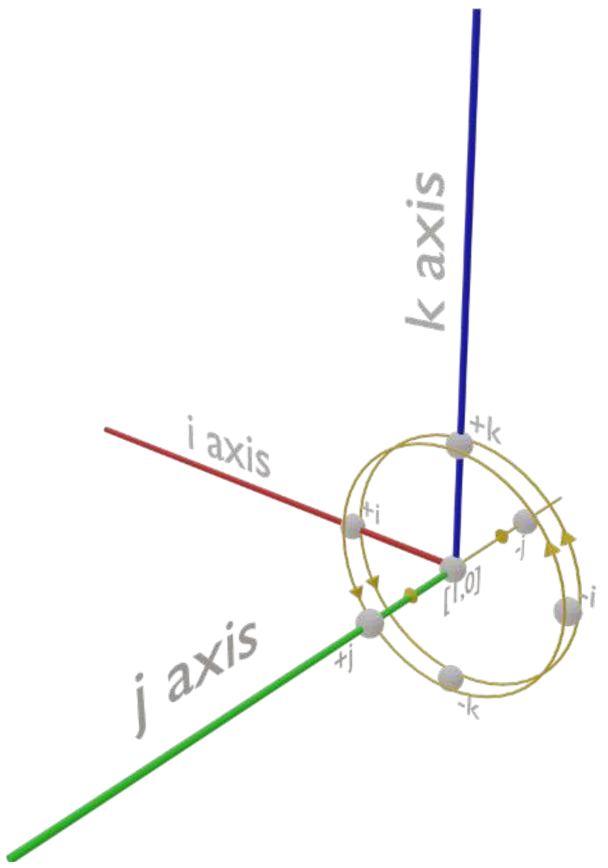


FIGURE 3.3: An example of a right multiplication of quaternion $[0, -j]$ in a 3D projection and left multiplication of quaternion $[0, j]$ (already shown in figure 3.2). According to equations (3.27) through (3.30), point i right-multiplicated by $-j$ goes to $-k$, point k right-multiplicated by $-j$ goes to i. Point 1, which lives in the origin of coordinate system, becomes $-j$. The total effect of the multiplication of the left quaternion and the right conjugate quaternion is canceled along the j axis, where the real value of the projection center was entangled. Conversely, the effect in jk is doubled.

Right multiplication with a conjugate $\boldsymbol{q_a^*}$, results in a similar behavior to left multiplication of $\boldsymbol{q_a}$. An example left multiplication with quaternion $[0, j]$ and right multiplication with quaternion $[0, -j]$ results in the same behavior in the ik. Do note that the behavior of real 1 affected by j and $-j$ effectively cancels out, as shown in figure 3.3. Also note that the rotation in the ik plane effectively doubles. A rotation is only applied in the 2D plane in ijk space around a unit vector in ijk space. Quaternion $\boldsymbol{s_{px}}$ is a pure quaternion, what is given in equation (3.39). The resulting quaternion $\boldsymbol{s_{p'x}}$ after rotation is also a pure quaternion. That means the equation (3.40) is a function that $\mathbb{R}^3 \to \mathbb{R}^3$. To construct a quaternion that will result in rotation of angle θ around normalized vector $\boldsymbol{v}$; $\|\boldsymbol{v}\| = 1$:

$$\mathbf{q_a} = \cos(\theta/2) + \sin(\theta/2)(v_x i + v_y j + v_z k) \tag{3.41}$$

This clearly shows the significance of division by two in the equation (3.41). Equation (3.41) is derived from a generalized polar representation of complex number. A complex number has one imaginary unit that is perpendicular to the real axis. A quaternion has three imaginary (vector) units which are simultaneously perpendicular to the real axis. Quaternions are effective means of representing rotation. They are minimal and complete representations of 3D rotation prevalent in game development and animation. They enable straightforward implementation of interpolation from one rotation to another. Finally, they are memory and computationally efficient and can be used to minimize throughput in applications where 3D rotation is measured or transferred, e.g. IMU sensors, or Central Processing Unit (CPU)-Graphics Processing Unit (GPU) communication in computer graphics.

Unit quaternions under multiplication are a Lie group S^3 and has corresponding Lie algebra $\mathfrak{s}^3$[115]. Unit quaternions group S^3 under multiplication has following group properties:

- identity element is quaternion number $[1, 0]$,
- operator is multiplication given with equation (3.32),
- inverse element is conjugate given with equation (3.37).

The Lie algebra are all pure quaternions : $[0, \boldsymbol{u}\phi] \in \mathfrak{s}^3$ where $\|\boldsymbol{u}\| = 1$. The exponential of Lie algebra member $[0, \boldsymbol{u}\phi]$ gives the unit quaterion as shown in equation (3.42).

$$\exp([0, \boldsymbol{u}\phi]) = \cos(\phi) + \boldsymbol{u}\sin(\phi); \exp([0, \boldsymbol{u}\phi]) \in S^3 \tag{3.42}$$

Hat operator allows to convert a rotation vector $\boldsymbol{\theta} \in \mathbb{R}^3$ to $\mathfrak{s}^3$. Such vector $\boldsymbol{\theta}$ is isomorphic to Lie algebra member:

$$(\boldsymbol{\theta})^\wedge = [0, \boldsymbol{\theta}/2] \tag{3.43}$$

Note that substituting $[0, \boldsymbol{\theta}/2]$ with $(\boldsymbol{\theta})^\wedge$ in equation (3.42) results in equation (3.41).

Quaternion interpolation

Spherical Linear Interpolation (SLERP) is an excellent example for the application of quaternions [114]. It utilizes a property of Lie algebra and Lie groups. In general, SLERP is an interpolation of point movement along an arc. For the argument of 0, the interpolation is the starting point of the arc. For argument 1, it is the ending point of the arc. Equation (3.44) demonstrates a SLERP implementation for an unit quaternion.

$$\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = (\mathbf{q}_1\mathbf{q}_0^{-1})^t\mathbf{q}_0 \tag{3.44}$$

Function $\mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t)$ takes as input $\mathbf{q}_0$ as the start rotation, $\mathbf{q}_1$ as the end rotation. For $t = 0$ the function slerp returns $\mathbf{q}_0$, and for $t = 1$ the function returns $\mathbf{q}_1$. The initial rotation of $\mathbf{q_0}$ is left multiplied by an incremental rotation from $\mathbf{q_0}$ to $\mathbf{q_1}$, which is $\mathbf{q_0}\mathbf{q_1}^{-1}$. The unit quaternion to the power of t is expressed using exponential map (3.42) as shown in equation (3.45).

$$\boldsymbol{q}^t = \exp(\boldsymbol{u}\phi)^t = \exp(t\boldsymbol{u}\phi) = \cos(t\phi) + \boldsymbol{u}\sin(t\phi) \tag{3.45}$$

SLERP Python implementation with all necessary elements of quaternion algebra is shown in the listing 3.1. Functions `quat_mul` and `quat_conj` are implemented directly from equations (3.32) and (3.37). Function `quat_slerp` is a direct implementation of equation (3.44) and (3.45). Function `quat_exp_map` is implementation of equation (3.42). Function `quat_log_map` is a logarithmic map of S^1 that for given member of S^1 returns $\mathfrak{s}^1$.

```python
import numpy as np

def quat_len(q):
    return np.sqrt(q[0]*q[0] + q[1]*q[1] + q[2]*q[2] + q[3]*q[3])

def quat_norm(q):
    d = quat_len(q)
    return [q[0]/d, q[1]/d, q[2]/d, q[3]/d]

def quat_conj(q):
    return [q[0], -q[1],-q[2],-q[3]]

def quat_inv(q):
    l = quat_len (q)
    return [q[0]/l, -q[1]/l,-q[2]/l,-q[3]/l]

def quat_mul(q_a, q_b):
    [s_a, x_a, y_a, z_a] = q_a
    [s_b, x_b, y_b, z_b] = q_b
    q = [0,0,0,0]
    q[0] = s_a*s_b - x_a*x_b - y_a*y_b - z_a*z_b
    q[1] = s_a*x_b + x_a*s_b + y_a*z_b - z_a*y_b
    q[2] = s_a*y_b - x_a*z_b + y_a*s_b + z_a*x_b
    q[3] = s_a*z_b + x_a*y_b - y_a*x_b + z_a*s_b
    return np.array(q)

def quat_log_map(q):
    phi = np.arccos(q[0])
    sin_theta = np.sin(phi)
    u = [q[1]/sin_theta,q[2]/sin_theta,q[3]/sin_theta]
    return np.array([0,phi*u[0],phi*u[1],phi*u[2]])

def quat_exp_map(q):
    phi = quat_len(q)
    sin_phi = np.sin(phi)
    u = quat_norm(q)
    return np.array([np.cos(phi), sin_phi*u[1],sin_phi*u[2],sin_phi*u[3]])

def quat_slerp (q0,q1, t):
    from_q0_to_q1 = quat_mul(q1, quat_conj(q0))
    tan_from_q0_to_q1 = quat_log_map(from_q0_to_q1)
    temp = quat_exp_map(t*tan_from_q0_to_q1)
    return quat_mul(temp, q0)
```

LISTING 3.1: Python code for SLERP and necessary quaternion operations.

Proper Euler angles and Tait-Bryan angles

Proper Euler and Tait-Bryan angles are another way to parameterize a rotation of a body in a 3D space. They combine three sequentially applied simple rotations called intrinsic rotations around the known basis. There are twelve combinations of these sequences. A rotation sequence of 1-2-3 can serve as an example of this sequence. In this case, the first rotation takes around the X-axis (α), the second one around the Y-axis (β), and the last one around the Z-axis (γ). This type of sequence is called a Tait-Bryan angle (or non-proper Euler angle). All twelve combinations are divided into two groups:

- proper Euler angles: $1-2-1, 1-3-1, 2-1-2, 2-3-2, 3-2-3, 3-1-3,$
- Tait-Bryan angles: $1-2-3, 1-3-2, 2-3-1, 2-1-3, 3-1-2, 3-2-1.$

Note that proper Euler angles use only two axes in their intrinsic rotations. Euler angles cannot be thought of as charts on the manifold of $SO(3)$. They also perform mapping, but it is entirely different from $\mathfrak{so}(3)$. First of all, Euler angles are not a Lie algebra. Small angle values may correspond to large rotations in a 3D space and vice versa. Moreover, not all elements of $SO(3)$ can be reached by Euler angles.

For every sequence of Euler angles there are two particular rotations which suffer from singularity. Take the Euler sequence of 1-2-1. It suffers from singularity at the second angle being equal 0 or π. The rotation matrix for this sequence is written as follows:

$$\mathbf{R} = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_x(\gamma) \tag{3.46}$$

where:
$\mathbf{R}_x(\alpha)$ - rotation around X axis around by first angle (α),
$\mathbf{R}_y(\beta)$ - rotation around Y axis around by second angle (β),
$\mathbf{R}_x(\gamma)$ - rotation around X axis around by third angle (γ).
After substituting β with 0 or π in equation (3.46), the rotation degenerates to (3.47).

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha)\cos(\gamma) - \sin(\alpha)\sin(\gamma) & -\sin(\alpha)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ 0 & \sin(\alpha)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma) - \sin(\alpha)\sin(\gamma) \end{bmatrix} \tag{3.47}$$

This is a degenerated case. Utilizing sum formula for Sine and Cosine, equation (3.47) can be simplified to equation (3.48).

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) \\ 0 & \sin(\alpha+\gamma) & \cos(\alpha+\gamma) \end{bmatrix} \tag{3.48}$$

In equation (3.48) first α and the third γ rotation effectively affect the resulting rotation in the exact same way. What is more (3.48) results in rotation around 'X' axis. It is visible that this representation for this particular configuration has only one degree of freedom. This phenomenon occurs both in case of Euler angles and Tait-Bryan angles. In the former, this effect appears when the second rotation angle is 0 or 180 degrees. In the latter, it appears for 90 or -90 degrees. This phenomenon has multiple interesting effects in engineering. One of them is the so-called gimbal lock. Assume a gyroscopic system where a rotor is attached to three nested gimbals and the housing is attached to a tracked body. The rotating body can affect those gimbals and keep the spinning gyro in the same configuration as the reference frame due to the law of preservation of angular momentum. The system can track the current orientation of the body by measuring the angles of every gimbal. Depending on the construction, which dictates the sequence of Euler angles, there is a possibility to configure the tracked body so that two gimbals align and further tracking 3D orientation is impossible. A locked gimbal will track 2D orientation. Fortunately modern optical and MEMS gyroscopes do not suffer from gimbal lock. Mechanical gyroscopes need to utilize redundancy of fourth axis, or need to be designed and operated carefully. Example of such careful operation could be found in transcript of Apollo mission, where mechanical gimbal was used [3].

Rigid body transformations in $SE(3)$ and their Lie algebra

$SO(3)$ only represents the rotations of a rigid body. To define a full 6-DOF pose of the rigid body, which is given by equation (3.49), $SE(3)$ space needs to be used.

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \in SE(3); \mathbf{R} \in SO(3) \tag{3.49}$$

$SE(3)$ is a Lie group that consists of two elements:

- the translation part $\mathbf{t} \in \mathbb{R}^3$ that is a trivial Lie group that itself is its Lie algebra,
- the rotational part $\mathbf{R} \in SO(3)$.

Transitional part of $SE(3)$ is a trivial Lie group under addition. Said Lie groups has manifolds that are planes thus, the Lie algebras are identical. For the particular vector group $\mathbb{R}^3$ under ordinary addition exponential mapping is of Lie algebra element $\boldsymbol{v}$ is $\boldsymbol{v}$ what is shown in. The same is true for logarithmic map what is shown in [115].

3.2.2 Motion model and smoothness

In this dissertation trajectory is understood as a set of consecutive 6-DOF poses. It can be represented in the form of a graph. It is a common practice to use a trajectory derived from odometry readings as a motion model. The motion model is built based on the set of relative poses described in section Relative Pose. Such motion model contributes constraints into graph-SLAM. For this reason it can be stated that trajectory is a crucial component for mobile mapping application. Moreover, consecutive relative poses can be used for building smoothness constraints. This is described in detail in section Smoothness.

Relative Pose

Two consecutive poses have a transformation (3.50) which lives in $SE(3)$ with $\boldsymbol{T}_{t-1}$ and $\boldsymbol{T}_t$. Poses at time t and $t-1$ are to be incrementally optimized by the solver. Thus, to incorporate the knowledge that is carried by the relative pose of $\boldsymbol{T}_{t-1}$ and $\boldsymbol{T}_t$, the following observation has to be added to the system.

$$\boldsymbol{T}_{(t-1)\to t} = {\boldsymbol{T}_{t-1}}^{-1}\boldsymbol{T}_t \tag{3.50}$$

Optimized poses are parameterized with $\boldsymbol{\sigma}_{t-1}^{\wedge} \in \mathfrak{se}(3)$ for time t and $\boldsymbol{\sigma}_{t}^{\wedge} \in \mathfrak{se}(3)$ and for time $t-1$.

Therefore, to keep the relative pose constraint, the equation (3.51) needs to be minimized.

$$\underset{\boldsymbol{\sigma}_{t-1},\boldsymbol{\sigma}_t}{\arg\min} \left\| \left(\log\left(\boldsymbol{T}_{(t-1)\to t}\right)\right)^{\vee} - \left(\log\left(\left(\exp(\boldsymbol{\sigma}_{t-1}{}^{\wedge})\right)^{-1}\exp(\boldsymbol{\sigma}_t{}^{\wedge})\right)\right)^{\vee} \right\| \tag{3.51}$$

Smoothness

Three consecutive transformations belong in $SE(3)$ with $\boldsymbol{T}_{t-1}$, $\boldsymbol{T}_t$, $\boldsymbol{T}_{t+1}$. It is assumed that the sampling of the trajectory is constant and the platform's velocity is locally constant. Based on this assumption, the relative pose from time moment $t-1$ to t should be equal to the relative pose from time moment t to $t+1$. The first relative pose is given with equation (3.52) and the second with (3.53). The final equation which needs to be minimized to maintain smoothness is given with (3.54).

$$\boldsymbol{T}_{(t)\to(t+1)} = \boldsymbol{T}_t{}^{-1}\boldsymbol{T}_{t+1} \tag{3.52}$$

$$\boldsymbol{T}_{(t-1)\to t} = \boldsymbol{T}_{t-1}{}^{-1}\boldsymbol{T}_t \tag{3.53}$$

Optimized poses are parameterized with $\boldsymbol{\sigma}_{t-1}^{\wedge} \in \mathfrak{se}(3)$ for time moment $t-1$, $\boldsymbol{\sigma}_{t}^{\wedge} \in \mathfrak{se}(3)$ for time moment t, $\boldsymbol{\sigma}_{t+1}^{\wedge} \in \mathfrak{se}(3)$ for time moment $t+1$. Therefore, to keep relative pose constraint, equation (3.54) needs to be minimized.

$$\underset{\boldsymbol{\sigma}_{t-1},\boldsymbol{\sigma}_t,\boldsymbol{\sigma}_{t+1}}{\arg\min} \left\| \left[\log\left(\left(\exp(\boldsymbol{\sigma}_{t-1}{}^{\wedge})\right)^{-1}\exp(\boldsymbol{\sigma}_t{}^{\wedge})\right)\right]^{\vee} - \left[\log\left(\left(\exp(\boldsymbol{\sigma}_t^{\wedge})\right)^{-1}\exp(\boldsymbol{\sigma}_{t+1}{}^{\wedge})\right)\right]^{\vee} \right\| \tag{3.54}$$

3.3 SLAM

SLAM is a vital aspect of the robotic system that enables robotic agents to find an optimal estimation of environment map and taken trajectory [138]. SLAM is formulated in many papers [55][73] and books [124] as probabilistic problem. State of the robotic system is described with multiple random variables:

$$\mathbf{x}_{1:T} = \{\mathbf{x}_1, ..., \mathbf{x}_T\} \tag{3.55}$$

A robot moving in an environment receives consecutive odometry readings $\mathbf{u}_{1:T}$ and set of K perception measurements $\mathbf{z}_{1:K}$.

$$\mathbf{u}_{1:T} = \{\mathbf{u}_1, ..., \mathbf{u}_T\} \tag{3.56}$$

$$\mathbf{z}_{1:T} = \{\mathbf{z}_1, ..., \mathbf{z}_K\} \tag{3.57}$$

The SLAM problem is finding a posterior probability of robot's trajectory $\mathbf{x}_{1:T}$, map $\mathbf{m}$, calibration of system $\mathbf{c}$ knowing a set of measurements $\mathbf{z}_{1:T}$, odometry $\mathbf{u}_{1:T}$ and initial pose of the system $\mathbf{x}_0$.

$$p(\mathbf{x}_{1:T}, \mathbf{m}, \mathbf{c} | \mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0) \tag{3.58}$$

This representation is universal. Robot poses can be represented as $SE(3)$ or $SE(2)$, depending on the application. Maps can be represented in various ways: as spatial landmarks, [86][35], a point cloud, or a dense 2D grid map. A Probability Distribution Function (PDF) of multivariate Gaussian distribution is given with equation (3.59).

$$p(\mathbf{x}) = \det(2\pi\mathbf{\Sigma})^{1/2} \exp\left\{ -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\intercal}\mathbf{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) \right\} \tag{3.59}$$

where information matrix is given with $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$, the expected value with $\boldsymbol{\mu}$, and the current value with $\mathbf{x}$. Bayes theorem enables probability inference with equation (3.60):

$$p(\mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m} | \mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0) = \frac{p(\mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0 | \mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m}) p(\mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m})}{p(\mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0)} \tag{3.60}$$

$p(\mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0)$ in equation (3.60) is a constant normalization term. Term $p(\mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0 | \mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m})$ expresses the probability of observed measurement $\mathbf{z}_{1:K}$, odometry $\mathbf{u}_{1:T}$ and initial pose $\mathbf{x}_0$ with given hidden variables (state). Equation (3.60) can be factorized with omitting normalization terms giving equation (3.61).

$$p(\mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m} | \mathbf{z}_{1:K}, \mathbf{u}_{1:T}, \mathbf{x}_0) \propto \prod_{k=1..K} p(\mathbf{z}_k | \mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m}) \tag{3.61}$$

Equation (3.61) shows that probability of a hidden variables is proportional to the measurement given value of that hidden variable. In later SLAM formulation it is taken into consideration. To find a probability distribution of measurement $\mathbf{z}_k$ given state $\mathbf{x}$ the equation (3.59) has to be substituted. The expected value $\boldsymbol{\mu}$ in that case is the expected measurement $\hat{\mathbf{z}}_k$. The result of this substitution is shown in equation (3.62). Note that only the exponential part of multivariate Gaussian distribution density function is present in equation (3.62).

$$p(\mathbf{z}_k | \mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m}) \propto \exp(-(\mathbf{z}_k - \hat{\mathbf{z}}_k)^{\intercal}\mathbf{\Omega}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k)) \tag{3.62}$$

Finding the most probable state $\mathbf{X}^*$ is the task of finding such $\mathbf{X}$ that maximizes the probability given in equation (3.63). The state $\mathbf{X}$ is trajectory $\mathbf{x}_{1:T}$, calibration $\mathbf{c}$ and map $\mathbf{m}$.

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \prod_{k=1..K} p(\mathbf{z}_k | \mathbf{x}_{1:T}, \mathbf{c}, \mathbf{m}) = \arg\max_{\mathbf{X}} \prod_{k=1..K} \exp(-\frac{1}{2}(\mathbf{z}_k - \hat{\mathbf{z}}_k)^{\intercal}\mathbf{\Omega}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{3.63}$$

After taking logarithm, equation (3.63) has the form (3.64).

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_{k=1..K} \frac{1}{2}(\mathbf{z}_k - \hat{\mathbf{z}}_k)^{\intercal}\mathbf{\Omega}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{3.64}$$

Finally, the optimization problem has the form of equation (3.65).

$$\mathbf{X}^* = \underset{\mathbf{X}}{\arg\min} \sum_{k=1..K} \frac{1}{2}\mathbf{e}_k(\boldsymbol{X})^\intercal \boldsymbol{\Omega}_k \mathbf{e}_k(\boldsymbol{X}) \tag{3.65}$$

where $\mathbf{e}_k(\mathbf{X})$ is a difference between observation and prediction given by equation (3.66).

$$\mathbf{e}_k(\mathbf{X}) = \mathbf{z}_k - \hat{\mathbf{z}}_k \tag{3.66}$$

3.3.1 Factor graph

The SLAM problem can be structured as a factor graph, which is a bipartite graph with two type of nodes:

- variable nodes,
- factor nodes that relate a subset of variables.

The edges connect variables to factors. Such tool can easily model SLAM problems. The problem modeled in figure 3.4 shows localization of robot in unknown map $\mathbf{m}$.

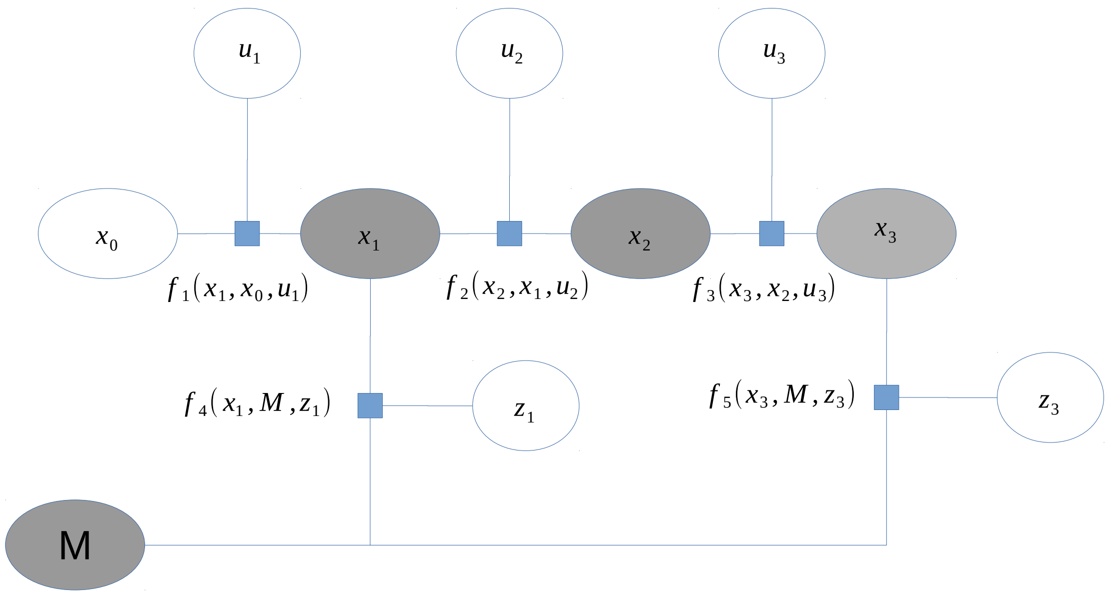


FIGURE 3.4: Sample SLAM problem modeled as a factor graph. Variable nodes are marked as circles and factors as squares. Gray color marks hidden variables.

Trajectory of robot is based on four poses, where the first one $\mathbf{x}_0$ is known and $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are hidden variables. Robot obtains odometry measurements as variables $\mathbf{u}_1$ to $\mathbf{u}_3$. Measurement from other sensors (e.g. laser range finder) are variables $\mathbf{z}_1$ and $\mathbf{z}_3$. Factors are functions that model conditional probability for given variables. Let factor node f_2 serves as example here. f_2 models conditional probability of particular pose of robot at time 2 given a pose at time 1 ($\mathbf{x}_1$), and relative movement measured by odometry of the robot traveling in time between moments 1 and 2 ($\mathbf{u}_2$). Solving this problem is to find the values of hidden variables ($\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ and $\mathbf{m}$) that maximize a total probability:

$$\underset{\mathbf{x}_{1..3},\mathbf{m}}{\arg\max} \prod_{1..5} f_i \tag{3.67}$$

Factor graphs are universal frameworks for state estimation. They can model other popular problems such as object tracking, structure from motion [33], calibration, or optimal control [120].

3.3.2 Graph SLAM

Graph SLAM is a well established method in the domain of mobile robotics [8][73][55]. This algorithm utilizes raw measurements of sensors and poses where:

- nodes are poses $\mathbf{x}_{1:T}$,
- edges (constraints) are measurements showing relative transformation between two poses.

Edges can be introduced by odometry measurement creating relative pose constraint; other classes of constraints can be introduced (based on observations). In the ideal and optimized system of constraints would not conflict, but can do so in a real-world scenario. Thus, a robust solution should be considered. The goal of Graph SLAM is to find the configuration of poses ($\mathbf{x}_{1:T}$) given all constraints that minimize overall residuals introduced by said constraints. Note that graph shown in 3.5 is a simple case.

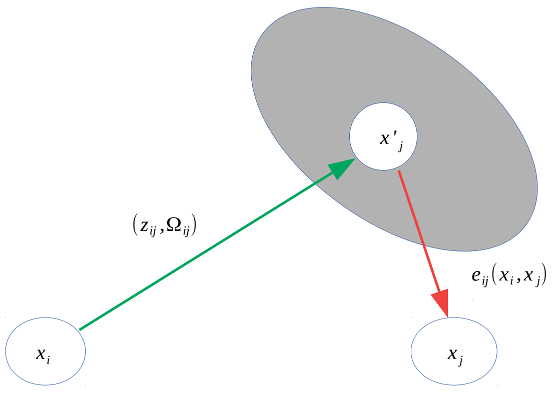


FIGURE 3.5: Figure is showing one edge of Graph SLAM. The $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are poses, which are hidden variables. There is an observation of pose $\boldsymbol{x}_j$ from pose $\boldsymbol{x}_i$ with value $\boldsymbol{z}_{ij}$ and information matrix $\boldsymbol{\Omega}_{ij}$. The observation allows to project an image $\boldsymbol{x}'_j$ of $\boldsymbol{x}_j$ respect to location $\boldsymbol{x}_i$ and observation $(\boldsymbol{z}_{ij}, \boldsymbol{\Omega}_{ij})$. Gray ellipse is confusion ellipse that visualizes information matrix. Error that this edge contributes is distance from image $\boldsymbol{x}'_j$ to location $\boldsymbol{x}_j$.

In a real-world system Graph SLAM can introduce non-sequential constraints, which are caused by loop-closure detection that introduces new constraints between poses located next to each other. Loop closure is desired, since it can greatly reduce accumulated registration errors. Solving Graph SLAM problem as pose graph is minimizing the following equation:

$$\arg\min_{\boldsymbol{X}} \mathbf{F}(\boldsymbol{X}) \tag{3.68}$$

where $\mathbf{F}(x)$ is the sum of all residuals:

$$\mathbf{F}(\boldsymbol{X}) = \sum_{\langle i,j\rangle \in C} \mathbf{e}_{ij}^{\intercal} \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \tag{3.69}$$

where $\mathbf{e}_{ij}$ is an error introduced by an edge connecting i-th and j-th pose and $\boldsymbol{\Omega}_{ij}$ is an information matrix (inverse of the covariance matrix). Figure 3.6 shows an example of graph SLAM problem before and after optimization.

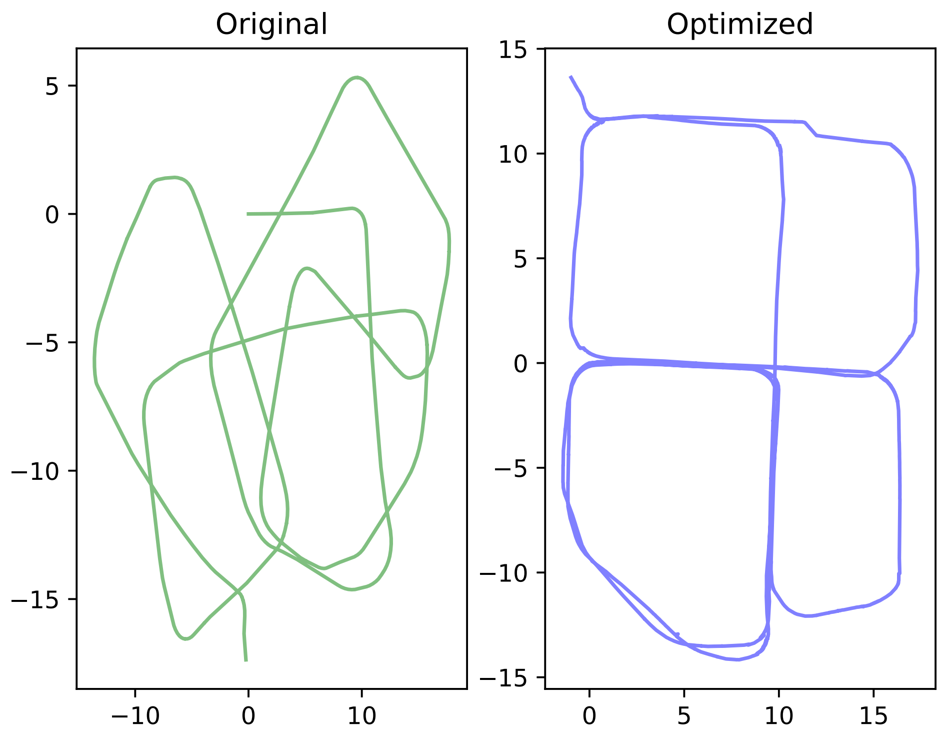


FIGURE 3.6: Graph SLAM problem before optimization and after optimization. Dataset is provided in [24] and software solution was used [2].

3.3.3 Bundle Adjustment

BA is a fairly similar problem to Graph SLAM. BA[73][2] is a problem that optimizes camera location in $SE(3)$, keypoint locations $\mathbb{R}^3$, and, optionally, the model of the camera itself (e.g., lens's focal length and lens's distortion coefficients). The BA problem does not have to be limited to only one camera model, as multiple cameras can be used. What is more, there is usually no temporal structure of cameras movement in the scene. These two factors differentiate BA from Graph SLAM. BA can be considered an essential part of every structure from motion solution [28]. The BA problem has an assumption of data correspondences. Usually, it comes from matching the smallest distance of descriptors of point of interest detected in the image. To make the selected optimization method converge successfully, a robust approach is required [73]. The comparison of such kernels can be found in [27]. In practice, many least-square solvers come with a set of such kernels, called also loss functors (see section 3.3.4). The second problem is the large size of the system. The selected solver needs to utilize the sparsity of the Jacobian matrix in the least square method. The third problem is sparse because cameras observe a small number of key points in the scene and it can be solved using the Schur complement of the Hessian matrix [44]. BA is a step in many stereovision pipelines that refines an initial calibration of the system [17][31].

3.3.4 Robust Nonlinear Least Squares

Both Graph SLAM and BA problems are solved using weighted least square method. This method finds a configuration of parameter $\boldsymbol{X}$ which minimizes the sum of squared error in an iterative manner. Error vector $\boldsymbol{e}(\boldsymbol{X})$ is given with equation(3.70).

$$\boldsymbol{e}(\boldsymbol{X}) = \boldsymbol{z} - \hat{\boldsymbol{z}}(\boldsymbol{X}) \tag{3.70}$$

where $\hat{\boldsymbol{z}}(\boldsymbol{X})$ is prediction, $\boldsymbol{z}$ is measurement. The optimized function is given with equation (3.71).

$$\underset{\boldsymbol{X}}{\arg\min} \frac{1}{2} \sum \boldsymbol{e}^{\intercal}(\boldsymbol{X}) \boldsymbol{W} \boldsymbol{e}(\boldsymbol{X}) \tag{3.71}$$

$\boldsymbol{e}$ is a vector of individual errors given with (3.72).

$$\boldsymbol{e}(\boldsymbol{X}) = \begin{bmatrix} e_1(\boldsymbol{X}) & e_2(\boldsymbol{X}) & \dots & e_n(\boldsymbol{X}) \end{bmatrix} \tag{3.72}$$

Matrix $\boldsymbol{W}$ is a weight matrix. The weight matrix contains information about probabilistic properties of the expected noise in measurement. Assuming uncorrelated noise in every dimension of error $\boldsymbol{e}(\boldsymbol{X})$ weight matrix $\boldsymbol{W}$ is given with (3.73).

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{bmatrix} \tag{3.73}$$

In equation (3.73) w_1,w_2 to w_n represent weights of individual error of the solution. I-th weight is given by an inverse of squared standard deviation (σ_i) of expected noise in i-th error :

$$w_i = \frac{1}{\sigma_i^2} \tag{3.74}$$

Thus, the equation (3.71) can be rewritten as a sum of every individual error:

$$\underset{\boldsymbol{X}}{\arg\min} \frac{1}{2} \sum_i \frac{e_i^2(\boldsymbol{X})}{\sigma_i^2} \tag{3.75}$$

The optimal $\boldsymbol{X}^*$ is found in the iterative manner by computing new, optimal solution $\boldsymbol{X}^*$ as (3.76).

$$\boldsymbol{X}^* = \boldsymbol{X} + \Delta\boldsymbol{X} \tag{3.76}$$

The $\Delta\boldsymbol{X}$ is an update step and $\boldsymbol{X}$ is the solution from last iteration. For the initial step $\boldsymbol{X}$ is an initial guess. In this thesis mainly the LM method is used. The update step in LM method is given with equation (3.77) [55].

$$\left(\boldsymbol{J}^\intercal \boldsymbol{W} \boldsymbol{J} + \lambda \boldsymbol{I}\right) \Delta\boldsymbol{X} = -\boldsymbol{J}^\intercal \boldsymbol{W} \boldsymbol{e}(\boldsymbol{X}) \tag{3.77}$$

where: λ is damping factor. $\boldsymbol{J}$ is Jacobian matrix of function $\boldsymbol{e}(\boldsymbol{X})$ given with equation (3.78).

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial x_1} & \cdots & \frac{\partial \mathbf{e}}{\partial x_n} \end{bmatrix} \tag{3.78}$$

The damping factor modifies the behavior of the optimization process. LM implementations set λ to large values, so the algorithm performs small updates in the direction of the steepest descent. If the step succeeds with the improvement of the cost function, the λ is decreased. With smaller λ, LM behaves like Gauss-Newton, which results in faster convergence [48]. The derivation of update step for LM and comparison to other popular algorithms, such as Gauss-Newton and gradient descent, can be found in [48]. In the following chapters, different software solutions are used to optimize non-linear least squares. In the case of complicated cost functions containing multiple chains of transformations Ceres, a library for the general approach, was used [2]. For structured, graph problems such SLAM the choice was specialized solutions: GTSAM [49] and g2o [73].

LM and Gauss-Newton methods are sensitive to outliers. Outliers are individual residual errors that does not follow Gaussian distribution. In SLAM an outlier can be introduced by a loop closure which is wrong or non-existing in a real environment. In BA outliers can be introduced by corrupted data association. An example of such distribution is presented in figure 3.7. The plot represents a histogram of residuals that were introduced by edges in sample Graph SLAM. The presented distribution does not follow the Gaussian distribution. To avoid such a situation in which a few outliers alter the results, robustifying kernels were introduced. Robustifying kernel is a non-linear function ρ that replaces quadratic function from equation (3.79) resulting in (3.80).

$$\arg\min_{\boldsymbol{X}} \sum_i \frac{1}{2} e_i^2(\boldsymbol{X}) \tag{3.79}$$

$$\arg\min_{\boldsymbol{X}} \sum_i \rho\left(e_i(\boldsymbol{X})\right) \tag{3.80}$$

where: $e_i(\boldsymbol{X})$ is individual i-th error.
Function ρ should be symmetric, positive define with a single, unique minimum at zero. The increase of function ρ should be smaller than parabolic. Iteratively Reweighted Least Squares (IRLS) is a method used to implement robust non linear least squares [135]. The IRLS introduces one more step to classical non-linear least square whose

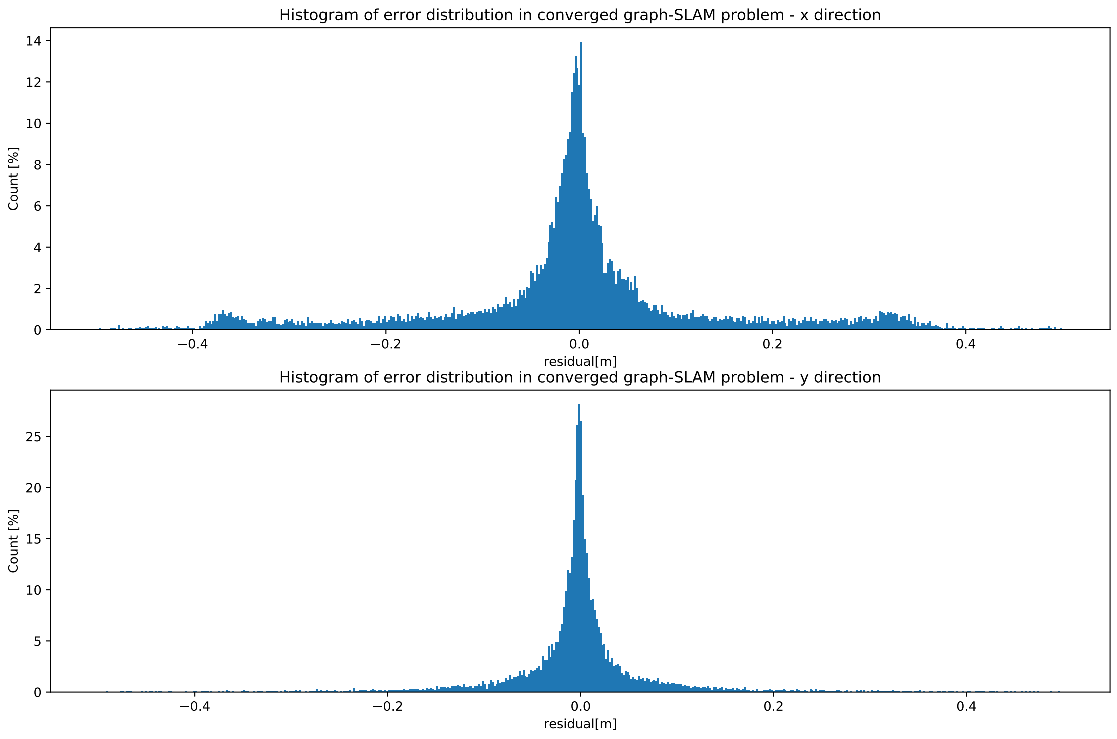


FIGURE 3.7: The distribution of translation error for axes 'X' and 'Y' in residuals in Graph SLAM problem shown in figure 3.6. Both of histograms presents a bell-shaped distribution. Those are symmetric, but suffer from excess kurtosis. The top distribution has kurtosis 3.76 and bottom 12.77. The high kurtosis is caused by large number of outliers and very long and fat tails. To summarize, those distributions are not Gaussian.

purpose is to find weights of every residual - w_{ri}.

$$\arg\min_{\boldsymbol{X}} \sum_i w_{ri} e_i^2(\boldsymbol{X}) \tag{3.81}$$

Formula for w_{ri} can be obtained from relations of IRLS (3.80) and weighted least square (3.81) optimization problem by comparing respective gradients at neighborhood of the solution for i-th residual. Gradient of i-th error for weighted least squares (3.81) is given with :

$$\frac{1}{2}\frac{\partial\left(w_{ri}e_i^2(\boldsymbol{X})\right)}{\partial \boldsymbol{X}} = w_{ri}e_i(\boldsymbol{X})\frac{\partial e_i(\boldsymbol{X})}{\partial \boldsymbol{X}} \tag{3.82}$$

Gradient of i-th error for robust least squares (3.80) is given with :

$$\frac{\partial\left(\rho(e_i(\boldsymbol{X}))\right)}{\partial \boldsymbol{X}} = \rho'(e_i(\boldsymbol{X}))\frac{\partial e_i(\boldsymbol{X})}{\partial \boldsymbol{X}} \tag{3.83}$$

Assuming that gradients (3.82) and (3.83) are going to zero at the optimum, w_{ri} is given with equation (3.84) [26].

$$w_{ri} = \frac{1}{e_i(\boldsymbol{X})}\rho'(e_i(\boldsymbol{X})) \tag{3.84}$$

where $\rho'(e_i(\boldsymbol{X}))$ is first order derivative with respect to $e_i(\boldsymbol{X})$.
Finally, by substituting found w_{ri} to equation (3.81), equation (3.85) is obtained. Equation (3.85) is the sum minimized by the IRLS. Weight w_{ri} is obtained from $e_i^{(k-1)}$ which is error value from previous iteration of IRLS algorithm.

$$\arg\min_{\boldsymbol{X}} \sum_i \frac{1}{e_i^{(k-1)}}\rho'\left(e_i^{(k-1)}\right)e_i^2(\boldsymbol{X}) \tag{3.85}$$

The IRLS implementation solves weighted least square problem with taking into consideration the error values obtained in previous iteration of algorithm. The values of the error from previous iteration are used to compute a set of new weights for the current iteration. Initially the weights are set to one. Note that IRLS problem can utilize noise model. Assuming that noise is independent for every error $e_i(\boldsymbol{X})$ and given with standard deviation σ_i such IRLS problem is given with (3.86).

$$\arg\min_{\boldsymbol{X}} \sum_i \frac{\sigma_i}{e_i^{(k-1)}}\rho'\left(\frac{e_i^{(k-1)}}{\sigma_i}\right)\frac{e_i^2(\boldsymbol{X})}{\sigma_i^2} \tag{3.86}$$

Note that if quadratic function was substituted as $\rho(e) = \frac{e^2}{2}$ in equation (3.86) the equation (3.86) would be identical to equation (3.75).

There are multiple robustifying kernels used for the implementation of the robust least-squares algorithm [135]. The most popular robustifying kernels and their weight impact are shown in figure 3.8. The blue plot is standard least-square implementation, the weight for every residuum value is always one. Other functions modify the weight in such a way so that only the residuals close to zero have the significant impact on the solution. The Huber kernel, which is a popular design choice for a robust kernel, is worth noticing here 3.8. Similarly to least square algorithm, Huber kernel is quadratic, like standard least square, but only in the thresholded region close to zero. Outside the thresholded region, the function is linear, which reduces the impact of outliers. The

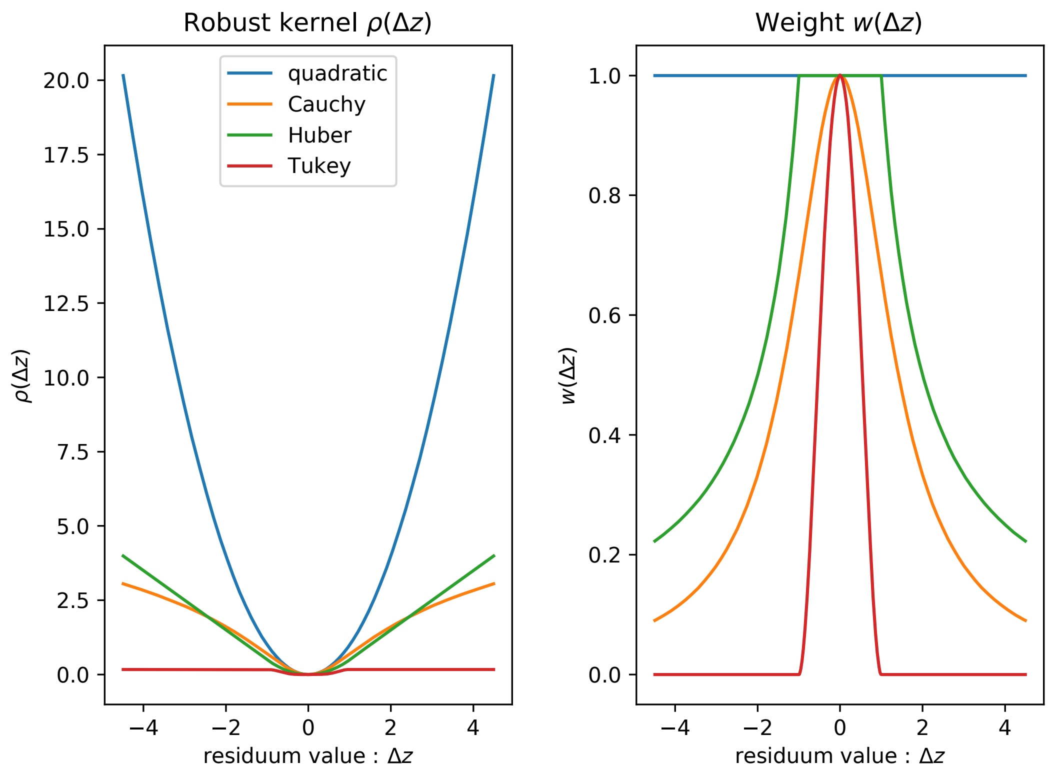


FIGURE 3.8: Robust kernel function and weight in function of residual value.

chosen robustifying kernel and its parameters need to fit the expected distribution of residual values. Automatic solutions for adaptive robust kernel is a topic which has recently become a subject of numerous research [6][26].

3.3.5 Map and trajectory accuracy assessment

The accuracy assessment of the robotic mapping system is a non-trivial task. Access to the ground-truth data is always an issue. It is evident in the literature that for the sake of benchmarks state-of-the-art data sets and methodologies were created. The first such example is presented in work [116]. Data sets presented in this work consist of measurements from RGBD cameras and high-framerate 6-DOF trajectory provided by a motion capture setup. The recovered trajectory estimated by SLAM or localization algorithm can be graded using two metrics: Relative Pose Error (RPE) and ATE. First one is sufficient for evaluating the difference between estimated and desired motion. It is used to assess the front-end of the SLAM algorithm, scan matching, optical, or even wheel odometry. It can show introduced drift or loop closure accuracy over a fixed time interval Δ.

$$\boldsymbol{E}_i = \left(\boldsymbol{Q}_i^{-1}\boldsymbol{Q}_{i+\Delta}\right)^{-1}\left(\boldsymbol{P}_i^{-1}\boldsymbol{P}_{i+\Delta}\right) \tag{3.87}$$

The equation (3.87) returns drift value in time moment i as $\boldsymbol{E}_i \in SE(3)$ assuming Δ as fixed time interval. $\boldsymbol{P}_1, .., \boldsymbol{P}_n \in SE(3)$ is estimated trajectory and $\boldsymbol{Q}_1, ..., \boldsymbol{Q}_n \in SE(3)$ is ground truth trajectory. To obtain scalar measurement of RPE, RMSE over translation part can be computed by equation (3.88).

$$RPE_{1:n}^{trans,\Delta} = RMSE(\boldsymbol{E}_{1:n}, \Delta) = \left(\frac{1}{m} \sum_{i=1}^{m} || \operatorname{trans}(\boldsymbol{E}_i) ||^2 \right)^{0.5} \tag{3.88}$$

where : $m = n - \Delta$. Alternatively a rotational RPE is formulated as mean of angular errors [101] :

$$RPE_{1:n}^{rot,\Delta} = \frac{1}{m} \sum_{i=1}^{m} \angle \operatorname{rot}(\mathbf{E}_i) \tag{3.89}$$

where : $\angle \operatorname{rot}(\boldsymbol{E}_i)$ recovers angle between origin and a rotation part of $\boldsymbol{E}_i$ as in equation (3.90). $\boldsymbol{R}_i$ is three by three upper left sub matrix of $\boldsymbol{E}_i$.

$$\angle \operatorname{rot}(\boldsymbol{R}_i) = \arccos \left(\frac{\operatorname{Tr}(\boldsymbol{R}_i) - 1}{2} \right) \tag{3.90}$$

The second method, ATE, involves aligning the estimated trajectory with ground truth. ATE metric is usable for the assessment of the global consistency of SLAM solution. The estimated trajectory $\boldsymbol{Q}_1, ..., \boldsymbol{Q}_n \in SE(3)$ and ground truth trajectory $\boldsymbol{P}_1, ..., \boldsymbol{P}_n \in SE(3)$ do not need to be in the same coordinate frame. The first step is to recover rigid body transformation using e.g. closed-form solution, Horn [62]. With recovered rigid body transformation $\boldsymbol{S}$ that aligns a ground truth and estimated trajectory, ATE error for the time moment i is formulated with equation (3.91).

$$\boldsymbol{F}_i := \boldsymbol{Q}_i^{-1} \boldsymbol{S} \boldsymbol{P}_i \tag{3.91}$$

To obtain a scalar measurement, RMSE over translation part over the whole trajectory is computed with equation (3.92).

$$RMSE(\boldsymbol{F}_{1:n}) = \left(\frac{1}{n} \sum_{i=1}^{n} || \operatorname{trans}(\boldsymbol{F}_i) ||^2 \right)^{0.5} \tag{3.92}$$

3.3.6 Ground truth data sources

Synthetic data sources

A synthetic dataset is beneficial for the evaluation of the methods of calibration. Blender [13] is used as a simulation tool the tests detailed later in the dissertation. Blender is a powerful and universal open-source software for modeling, animation and video production. An example render is shown in figure 3.9. Blender includes a modern, robust, and configurable path tracing engine, as well as Python Application Programming Interface (API), which is why both accurate ground truth depth and photo-realistic images can be obtained easily.

Due to this fact, many researchers [25][69][19][83] create synthetic data sources using Blender. It is often better suited for simulating depth ground truth than solutions such as Gazebo [70], Carla [36] and other [104][22], in which a rasterization real-time pipeline is used. Those solutions can produce acceptable results, but they are oriented for real-time or near real-time data simulation. Those provide quality of image that can be insufficient for many applications. The output resolution is limited by factors such as graphic card memory. Depth output is limited and nonlinear due to limitations of graphic card architecture. Rendered images has their dynamic range limited. That is caused by underlying technology like Unity [126] or Unreal [127]. These are real-time game engines optimized for fidelity and smooth animation with the highest possible frame rate. However, these solutions scale poorly; it is not feasible to run those engines in the cloud, headless, or containerized environment.

On the other hand, path tracing can produce floating-point ground truth images with unlimited dynamic range and floating-point depth. The advantage of such solution is a large number of camera models available. It is worth mentioning that path tracing can produce detailed maps that are not available by other techniques, e.g. normal maps with floating precision, optical flow, and segmentation masks. Some researchers decided to take available open source community-made short animation videos, and repurpose them as open data sets, for example 2012 *MPI Sintel* [19] for optical flow or 2016 *Scene Flow* [83], designed for training artificial neural network for disparity estimation. Finally, solutions like Blender scale very well due to the primary use case, i.e. massive rendering of animated videos.

Blender offers a plethora of features which are useful in generating test data. First of all, it utilises a number of camera models:

- standard pinhole model,
- orthographics camera,
- equirectangular,
- fish-eye.

In a path tracing or a ray tracing engine, every pixel of the simulated camera emits some simulated samples which traverse the scene. When sample-path collides with an object, the path tracing engine executes Bidirectional Scattering Distribution Function (BSDF). This probabilistic function simulates physical light scattering or traversing. The following sample travels until it meets the next object or light source. The results of those collisions are collected and used to yield the final color of the pixel.

This method can be considered a Monte-Carlo class algorithm [63] as its results are heavily dependent on the number of samples, complexity of the BSDF used, number of light sources and the scene itself. A poor number of samples causes rendering noise. Path tracing, real time ray tracing, light transport and rendering noise removal are all vast topics which have been widely researched in recent years.

The Cycles rendering engine allows for the creation of multiple extra channels called render layers:

- 32 bit floating point color,
- 32 bit depth - linear (Z-pass),
- material and objects index - a 16-bit integer that is output as a map,

- floating point normal vector 3D,
- floating point 2D pixel disparity to next frame (optical flow),
- number of other output that enables artists to post-process the render.

The output can be easily organized in multi-layer OpenEXR files which can be read by existing libraries in Python or C++. Multiple technologies of depth acquisition are available, e.g. an entirely passive system such as stereo pairs [31] or active LiDARs [78]. These depth acquisition sensors can have different FOV, which needs to be taken into consideration during the simulation.

The sensors used in the evaluation part of this thesis are a good example. The first one is Intel RealSense L515. It is a modern solid-state LiDAR which provides Extended Graphics Array (XGA) (1024x768) resolution up to 4 meters with FOV 70° x 55 °. Its output is very similar to stereo-vision, but it is obtained with TOF. Simulating such sensor requires:

- a perspective camera with FOV ° x 55°,
- output resolution 1024 x 768 pixels,
- export 'Z' and 'RGB'.

Another interesting measurement instrument is Velodyne VLP-16. It is a popular and affordable automotive-grade LiDAR that utilizes a rotating head (600 revolutions per minute) consisting of 16 pairs of laser range finders. This sensor can be simulated in a stationary or slow-moving situation with the given model:

- an equirectangular camera with FOV 16° x 360°,
- output resolution 16x3600 pixels,
- export 'Z'.

The simple simulation used for producing synthetic results to validate a calibration method is shown in figure 3.9. It consists of two simulated Intel L515 LiDARs rotating around the calibration plane with fiducial markers. RGB and depth image are rendered with a Blender-Cycles renderer.

Real data

Reliable and accurate data sources are used as a ground truth. Dominant data is GNSS and geo-referenced point clouds. Due to the usage of precise surveyor equipment and manual registration, the map derived from Terrestial Laser Scanning (TLS) has the accuracy of even up to 3 milliliters. The map used for verification in chapter 4.5 was obtained with Z+F Imager 5010 TLS, a device which has an even higher precision of 1 mm [64]. Other data sets used for benchmarking SLAM or localization algorithms with ground truth trajectories were obtained using a calibrated, high frame-rate motion capture system which retrieves the pose of the sensor moving through the scene [116].

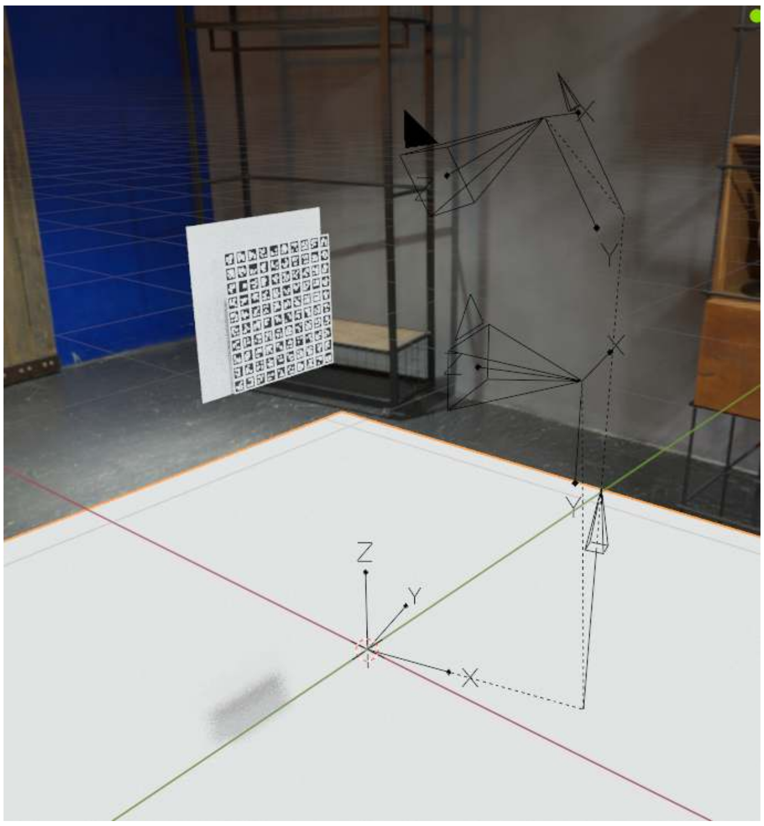


FIGURE 3.9: Simulated calibration scene for synthetic equivalent of system shown in figure 4.6. Two cameras RGBD cameras are marked. HDR map in background imitates real-world illumination.

3.4 LiDAR observation equations

3.4.1 Point to Point

There are two sets of points $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ with one to one correspondences. The size of pointclouds $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ are N. The registration problem expressed in equation (3.93) is to find $\boldsymbol{\sigma}_t^\wedge \in \mathfrak{se}(3)$ and $\boldsymbol{\sigma}_s^\wedge \in \mathfrak{se}(3)$ parameters which minimize overall distance between corresponding points in two registered point clouds:

$$\underset{\boldsymbol{\sigma}_t,\boldsymbol{\sigma}_s}{\arg\min} \sum_{i=0}^{N} \left\| \exp(\boldsymbol{\sigma}_t^\wedge)\boldsymbol{X}_t(i) - \exp(\boldsymbol{\sigma}_s^\wedge)\boldsymbol{X}_s(i) \right\| \tag{3.93}$$

where $\boldsymbol{X}_t(i)$ is an i-th point in the target pointcloud and $\boldsymbol{X}_s(i)$ is an i-th point in the source point cloud. The pose of point cloud $\boldsymbol{X}_s$ is parametrized with $\boldsymbol{\sigma}_s$, and the pose of point cloud $\boldsymbol{X}_t$ is parametrized with $\boldsymbol{\sigma}_t$. The point to point registration problem has a simple analytical Jacobian, which utilizes some properties of Lie algebra.

Jacobian matrix in SE(3) for point transformation

Point $\mathbf{p}$ is transformed via $[\mathbf{R}, \boldsymbol{t}]$ to its new location with equation (3.94).

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{3.94}$$

Partial derivatives of point $\mathbf{p}'$ in respect to parameters $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ are given with equation (3.95). $\boldsymbol{\rho}$ are parameters that are related to translation and $\boldsymbol{\sigma}$ are parameters related to rotation. The operator $[.]_\times$ builds a 3×3 skew-symmetric matrix from a vector containing 3 elements.

$$\begin{bmatrix} \frac{\delta(\mathbf{Rp}+\mathbf{t})}{\delta\boldsymbol{\rho}} & \frac{\delta(\mathbf{Rp}+\mathbf{t})}{\delta\boldsymbol{\sigma}} \end{bmatrix} = \begin{bmatrix} \frac{\delta x'}{\delta\rho_1} & \frac{\delta x'}{\delta\rho_2} & \frac{\delta x'}{\delta\rho_3} & \frac{\delta x'}{\delta\sigma_1} & \frac{\delta x'}{\delta\sigma_2} & \frac{\delta x'}{\delta\sigma_3} \\ \frac{\delta y'}{\delta\rho_1} & \frac{\delta y'}{\delta\rho_2} & \frac{\delta y'}{\delta\rho_3} & \frac{\delta y'}{\delta\sigma_1} & \frac{\delta y'}{\delta\sigma_2} & \frac{\delta y'}{\delta\sigma_3} \\ \frac{\delta z'}{\delta\rho_1} & \frac{\delta z'}{\delta\rho_2} & \frac{\delta z'}{\delta\rho_3} & \frac{\delta z'}{\delta\sigma_1} & \frac{\delta z'}{\delta\sigma_2} & \frac{\delta z'}{\delta\sigma_3} \end{bmatrix} \tag{3.95}$$

$$\mathbf{R} = \exp([\boldsymbol{\sigma}]_\times) \tag{3.96}$$

$$\frac{\delta f(X)}{\delta\varphi} = \lim_{\varphi\to 0} \frac{f(X \oplus \varphi) - f(X)}{\varphi} \tag{3.97}$$

$$\begin{aligned} \frac{\delta(\mathbf{Rp})_\mathbf{r}}{\delta\boldsymbol{\sigma}} &= \lim_{\boldsymbol{\sigma}\to 0} \frac{\mathbf{R}\exp([\boldsymbol{\sigma}]_\times)\mathbf{p} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ &\lim_{\boldsymbol{\sigma}\to 0} \frac{\mathbf{R}(\mathbf{I} + [\boldsymbol{\sigma}]_\times)\mathbf{p} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ &\lim_{\boldsymbol{\sigma}\to 0} \frac{\mathbf{Rp} + \mathbf{R}([\boldsymbol{\sigma}]_\times)\mathbf{p} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ \lim_{\boldsymbol{\sigma}\to 0} \frac{\mathbf{R}[\boldsymbol{\sigma}]_\times\mathbf{p}}{\boldsymbol{\sigma}} &= \lim_{\boldsymbol{\sigma}\to 0} \frac{-\mathbf{R}[\mathbf{p}]_\times\boldsymbol{\sigma}}{\boldsymbol{\sigma}} = -\mathbf{R}[\mathbf{p}]_\times \end{aligned} \tag{3.98}$$

The Jacobian for rotation in (3.98) is a right Jacobian, general equation (3.97). It is compatible with right-$\oplus$ operator. Left Jacobian (general equation (3.99)) derivation is shown in (3.100).

$$\frac{\delta f(X)}{\delta\varphi} = \lim_{\varphi\to 0} \frac{f(\varphi \oplus X) - f(X)}{\varphi} \tag{3.99}$$

$$\begin{aligned} \frac{\delta(\mathbf{Rp})_\mathbf{l}}{\delta\boldsymbol{\sigma}} &= \lim_{\boldsymbol{\sigma}\to 0} \frac{\exp([\boldsymbol{\sigma}]_\times)\mathbf{Rp} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ &\lim_{\boldsymbol{\sigma}\to 0} \frac{(\mathbf{I} + ([\boldsymbol{\sigma}]_\times)\mathbf{Rp} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ &\lim_{\boldsymbol{\sigma}\to 0} \frac{\mathbf{Rp} + ([\boldsymbol{\sigma}]_\times)\mathbf{Rp} - \mathbf{Rp}}{\boldsymbol{\sigma}} = \\ \lim_{\boldsymbol{\sigma}\to 0} \frac{[\boldsymbol{\sigma}]_\times(\mathbf{Rp})}{\boldsymbol{\sigma}} &= \lim_{\boldsymbol{\sigma}\to 0} \frac{-[\mathbf{Rp}]_\times\boldsymbol{\sigma}}{\boldsymbol{\sigma}} = -[\mathbf{Rp}]_\times \end{aligned} \tag{3.100}$$

In derivations (3.98) and (3.100) two properties are taken into account [115]:

- $\exp(\boldsymbol{\sigma})^\wedge \approx \mathbf{I} + [\boldsymbol{\sigma}]_\times$ (the Lie algebra is tangent to manifold locally),
- $[\boldsymbol{a}]_\times \boldsymbol{b} = -[\boldsymbol{b}]_\times \boldsymbol{a}$ (the cross product is anti-commutative).

With equation (3.98) and (3.100) right three by three submatrix of Jacobian (3.95) was found. The found submatrix of Jacobian (3.95) represents the partial derivative respective to rotation $\boldsymbol{\sigma}$. To obtain the left block built from partial derivatives respective to translation $\boldsymbol{\rho}$ from (3.95), one must formulate a derivative equation respective to the translation. Translation is a trivial Lie group. $\oplus$ and $\ominus$ can be replaced with ordinary addition and subtraction. The equation (3.97) can be formulated for the right Jacobian of $\boldsymbol{\rho}$ (3.101).

$$\frac{\delta(\boldsymbol{Rp}+\boldsymbol{t})_{\mathbf{r}}}{\delta\boldsymbol{\rho}} = \lim_{\boldsymbol{\rho}\to 0} \frac{\boldsymbol{R}(\boldsymbol{p}+\boldsymbol{\rho})+\boldsymbol{t}-\mathbf{Rp}}{\boldsymbol{\rho}} = \lim_{\boldsymbol{\rho}\to 0} \frac{\boldsymbol{R\rho}+\boldsymbol{t}}{\boldsymbol{\rho}} = \mathbf{R} \tag{3.101}$$

The equation (3.99) can be formulated for the right Jacobian of $\boldsymbol{\rho}$ (3.102).

$$\frac{\delta(\boldsymbol{Rp}+\boldsymbol{t})_{\mathbf{l}}}{\delta\boldsymbol{\rho}} = \lim_{\boldsymbol{\rho}\to 0} \frac{(\mathbf{Rp}+\mathbf{t})+\boldsymbol{\rho}-\mathbf{Rp}}{\boldsymbol{\rho}} = \mathbf{I} \tag{3.102}$$

Finally, the right Jacobian of (3.94) is given with equation (3.103), and the left Jacobian is given with equation (3.104).

$$\begin{bmatrix} \frac{\delta(\mathbf{Rp}+\mathbf{t})_{\mathbf{r}}}{\delta\boldsymbol{\rho}} & \frac{\delta(\mathbf{Rp}+\mathbf{t})_{\mathbf{r}}}{\delta\boldsymbol{\sigma}} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}[\mathbf{p}]_\times \end{bmatrix} \tag{3.103}$$

$$\begin{bmatrix} \frac{\delta(\mathbf{Rp}+\mathbf{t})_{\mathbf{l}}}{\delta\boldsymbol{\rho}} & \frac{\delta(\mathbf{Rp}+\mathbf{t})_{\mathbf{l}}}{\delta\boldsymbol{\sigma}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -[\mathbf{Rp}]_\times \end{bmatrix} \tag{3.104}$$

Iterative Closest Point

The ICP algorithm [9] finds the transformation between the point clouds based on the measured distances between the nearest neighboring points. Searching for the nearest neighborhood is a well-known problem in computer graphics and robotics [129][12]. In this dissertation the nearest neighborhood search is implemented using Fast Library for Approximate Nearest Neighbors (FLANN). This allows for finding the nearest neighborhood after building the kd-tree. Thus, for each query point, its nearest neighbor is found and this pair is taken as an input for constructing a point to point observation equation. The two initial experiments were performed for the ICP investigation. The results of the first one are shown in figure 3.10. The goal of the investigation was to compare the impact of the chosen Jacobian calculations on the overall convergence. In case of the perturbation model Jacobian (3.103), the convergence is much better than for a Jacobian using symbolic calculations. It shows the advantage of the Lie algebra approach. The second experiment shown in figure 3.11 compares the convergence of the ICP algorithm for different Jacobians. Using a Lie algebra gives the fastest convergence. This observation leads to the decision that further investigation related to the automatization of the calibration process shall use the Lie algebra for optimization problems.

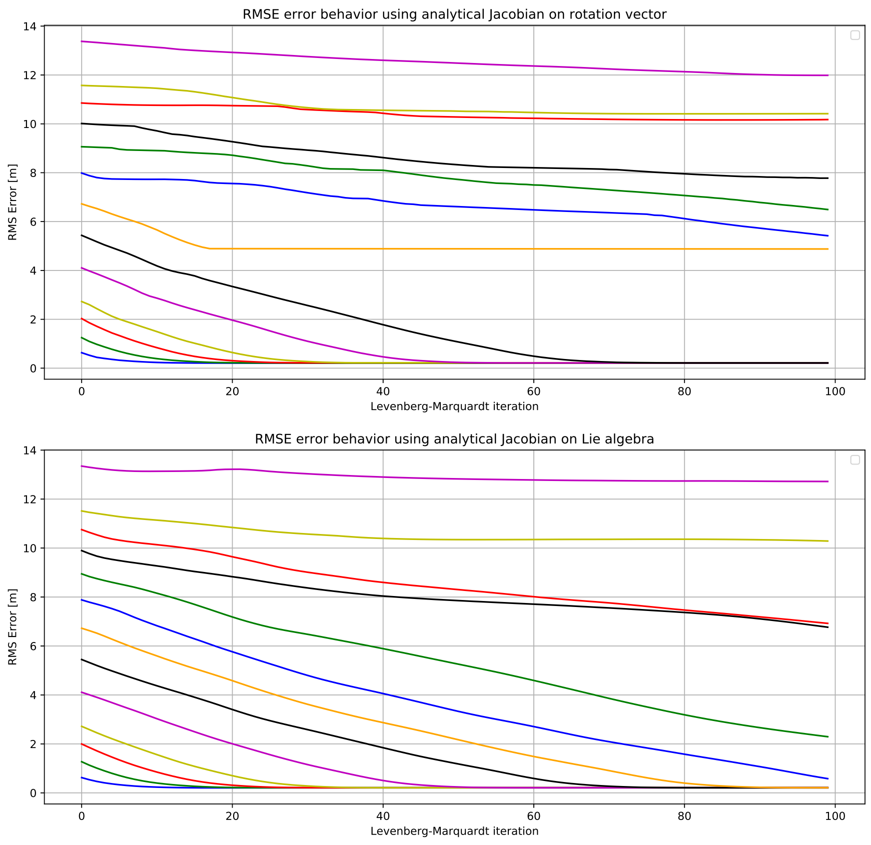


FIGURE 3.10: The plot of RMSE after a finished iteration of the ICP on a synthetic data-set for multiple initial condition. Various colors represent different initial errors. The violet plot, which fails to optimize, represents the largest initial error. The first method (top) uses a symbolically computed Jacobian using SymPy [84]. The second one (bottom) uses an analytical Jacobian on a Lie algebra with equation (3.103). Each initial error was introduced by increased rotation. Thus, both methods converge only for the limited initial error. For more significant errors, several iterations are too few. Otherwise, the optimization fails to converge at all. Both methods are sufficient, but the perturbation model Jacobian provides better stability and robustness.

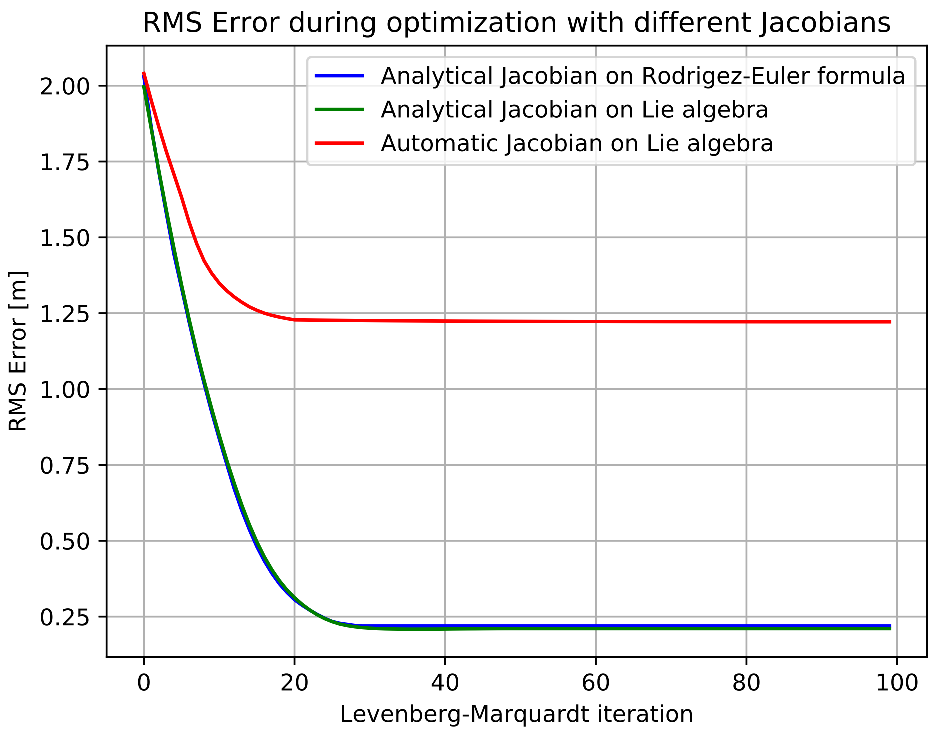


FIGURE 3.11: The plot of error after the finished iteration of the ICP on synthetic dataset. The convergence of Levenberg-Marquardt ICP is compared for different Jacobians. All these optimizers find a parameter which lives in $\mathfrak{se}(3)$ space. The red plot shows a RMSE during the optimization using an automatic Jacobian [2]. The green and blue show the RMSE during optimization using an analytical Jacobian. The first one is a symbolical differentiation of Rodrigues formula (3.18) using SymPy [84]. The second one is a Jacobian on a Lie algebra which is given in equation (3.103). The symbolic and analytical Jacobian on a Lie algebra gives similar results, while analytical on a Lie algebra is more computationally efficient.

3.5 Camera observation equations

3.5.1 Line to line in equirectangular image

This section describes novel approach to calibrate spherical camera using lines features. There are utilized properties of projective plane, a properties of Plücker coordinates [103]. Approach that utilizes line feature in SLAM are emerging [137][102][136]. Those solutions tends to provide robust optimization in structured man-made environments. In derivation of line to line observation equation in equirectangular image properties of a great circle in equirectangular are utilized [60]. Lines have multiple interesting properties in equirectangular images, which can be utilized in solving extrinsic calibration problems and camera alignment. A line which is observed by the equirectangular camera forms an arc in the output image. The line creates a plane that contains:

- a line,
- a projection center.

The last statement is true both for an equirectangular projection and a perspective projection because these two projections satisfy the collinearity condition. A line in a 3D scene is represented with Plücker coordinates [103]. A Plücker coordinates represent a line in a 3D space with two orthogonal vectors: $\boldsymbol{m}$ and $\boldsymbol{l}$, the first being a moment vector, and the latter a directional vector. The vectors $\boldsymbol{m}$ and $\boldsymbol{l}$ described line are shown in figure 3.12. The points $\boldsymbol{x}$ and $\boldsymbol{y}$ are lying on the same line, and are unit apart:

$$\|\boldsymbol{x} - \boldsymbol{y}\| = 1 \tag{3.105}$$

The directional vector $\boldsymbol{l}$ has unit length and can be obtained with:

$$\boldsymbol{l} = \boldsymbol{x} - \boldsymbol{y} \tag{3.106}$$

The moment vector $\boldsymbol{m}$ is defined as:

$$\boldsymbol{m} = \boldsymbol{x} \times \boldsymbol{y} \tag{3.107}$$

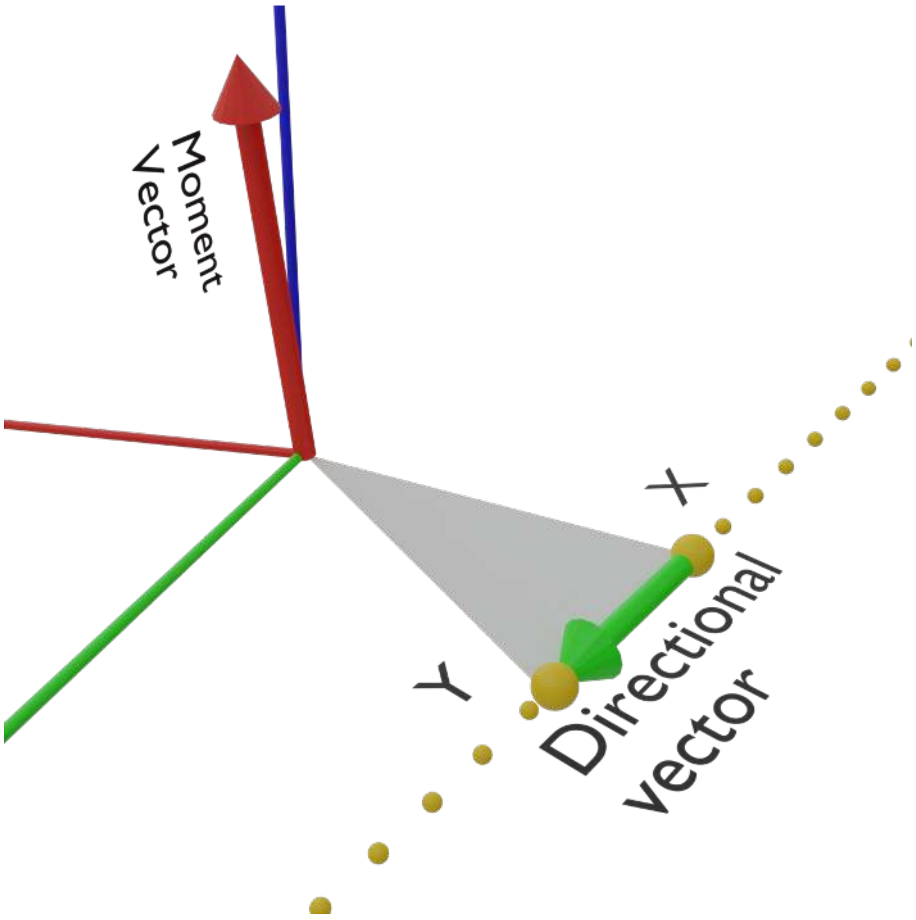


FIGURE 3.12: The line segment with (yellow) and vectors that creates Plücker coordinate. Moment vector $\boldsymbol{m}$ is marked in red and directional vector $\boldsymbol{l}$ is marked in green. Note that $\boldsymbol{m}$ length is two times $\boldsymbol{l}$ length. That due to fact of yellow line to be 2 units away from begin of coordinate system.

A line with Plücker coordinate is given:

$$\boldsymbol{L}^w = \begin{bmatrix} m_x \\ m_y \\ m_z \\ l_x \\ l_y \\ l_z \end{bmatrix} \tag{3.108}$$

This 6D representation of a 3D line $\boldsymbol{L}^w$ can be transformed from the world coordinate frame to a camera's coordinate $\boldsymbol{L}^c$ frame with equation (3.109). $\boldsymbol{R}_{wc} \in SO(3)$ represents rotation of the camera in world coordinate frame and $\boldsymbol{t}_{wc} \in \mathbb{R}^3$ represents translation of the camera in world coordinate frame [76]. Operator $[.]_\times$ is a skew symmetric matrix.

$$\boldsymbol{L}^c = \begin{bmatrix} \boldsymbol{m}_l \\ \boldsymbol{l}_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{wc}^\intercal & -\boldsymbol{R}_{wc}^\intercal[\boldsymbol{t}_{wc}]_\times \\ \boldsymbol{0}^{3x3} & \boldsymbol{R}_{wc}^\intercal \end{bmatrix} \boldsymbol{L}^w \tag{3.109}$$

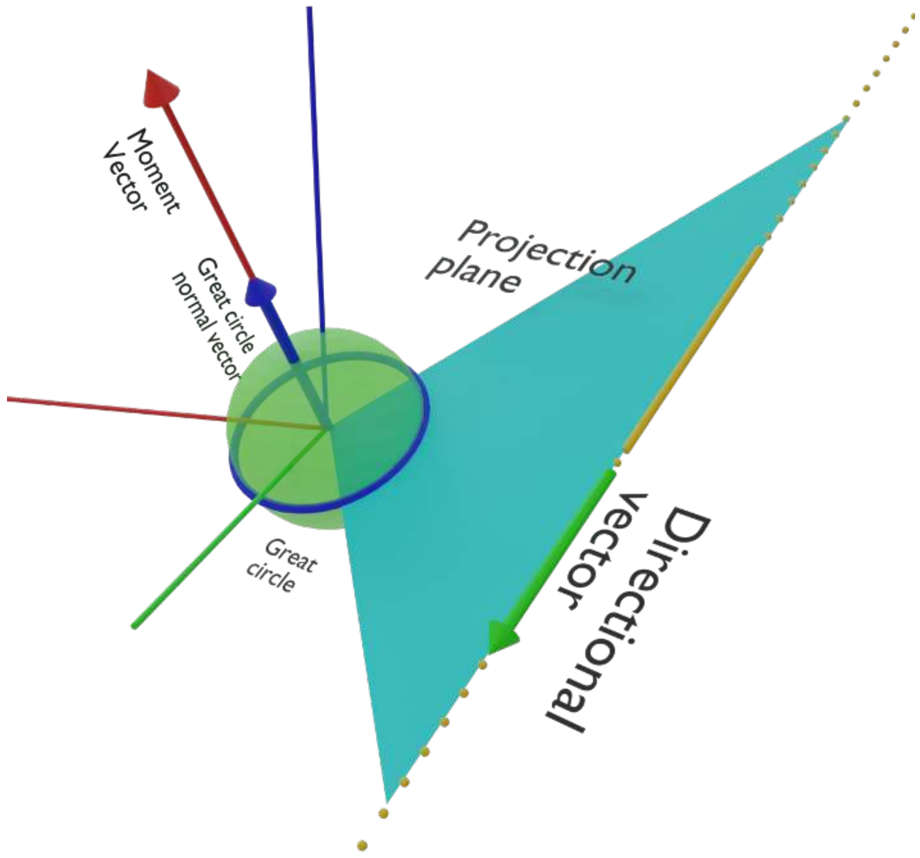


FIGURE 3.13: The line segment in a 3D space is marked in yellow. There are two vectors that create the Plücker coordinates: the moment is marked in red and the directional vector is marked in green. The projection plane spans on the center of the projection sphere (marked in green) and the observed line (yellow). This plane contains a great circle (marked in blue) of the projection sphere. A normal vector of the great circle is marked in blue.

Taking a normalized vector $\boldsymbol{n}_l = \boldsymbol{m}_l / \|\boldsymbol{m}_l\|$ from the local Plücker coordinate gives a normal vector for a plane which goes through the observed line, and the projection center of the camera in the camera's coordinate frame. The intersection of the plane expressed with normal vector $\boldsymbol{n}_l$ and a projection sphere gives a circle. It is a great circle if the plane contains the center of the sphere. It is always valid because the camera projection center, every point on the projection sphere, and their respective points in 3D of the observed feature are collinear. All major vectors and geometries are shown in drawing 3.13. Note that vector $\boldsymbol{n}_l$ and opposite $-\boldsymbol{n}_l$ give an identical great circle on the sphere. For the plane with normal vector $[a, b, c]$ and containing the origin, the equation (3.110) is satisfied.

$$\begin{bmatrix} a & b & c & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{3.110}$$

For the great circle with azimuth (α) and inclination (β), the equation (3.110) becomes (3.111).

$$\begin{bmatrix}\sin(\alpha)\sin(\beta) & -\cos(\alpha)\sin(\beta) & \cos(\beta) & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = 0 \tag{3.111}$$

The plane (given with equation (3.110)) is projected by equirectangular mapping on spherical images using equation (1.5). The result of the substitution is given in equation (3.112).

$$a\sin(\varphi)\sin(\vartheta) + b\sin(\vartheta)\cos(\varphi) + c\cos(\vartheta) = 0 \tag{3.112}$$

The equation (3.112) can be solved for longitude ϑ, resulting in:

$$\tan(\vartheta) = \frac{-c}{a\sin(\varphi) + b\cos(\varphi)} \tag{3.113}$$

Replacing a,b,c in equation (3.113) with (3.111) formula gives (3.114).

$$\tan(\vartheta) = \frac{-1}{\tan(\beta)(\sin(\alpha)\cos(\varphi) - \sin(\varphi)\cos(\alpha))} \tag{3.114}$$

Utilizing the sum of sine angles equation to equation (3.114) simplifies to equation (3.115)

$$\tan(\vartheta) = \frac{-1}{\tan(\beta)(\sin(\alpha - \varphi))} \tag{3.115}$$

In other words, every line visible in a spherical image can be parameterized with two angles: an azimuth α and an inclination β. The effect of those two angles on the projection of a line is shown in figure 3.14. Given a normal vector of a great circle as

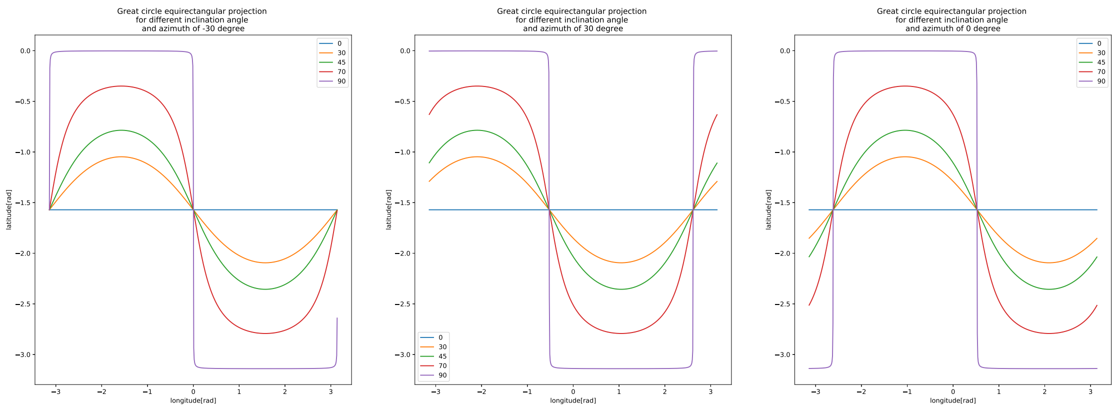


FIGURE 3.14: Great circle projection equation (3.115) for various values of inclination and azimuth.

$\boldsymbol{n}_l = [n_{xl}, n_{yl}, n_{zl}]$, the azimuth and the inclination can be found with the following equations:

$$\alpha = \arccos(n_{zl}) \tag{3.116}$$

$$\beta = -\arctan\left(\frac{n_{xl}}{n_{yl}}\right) \tag{3.117}$$

Curve fitting

For a given set of points in a spherical image which belongs to the same line, parameters α and β have to be found. A point with coordinates u_1 and v_1 in a spherical image can be converted to respective longitude ϑ_1 and latitude φ_1 with formula:

$$\vartheta_1 = \pi\frac{u_1 - 0.5c}{c} \tag{3.118}$$

$$\varphi_1 = -\pi\frac{v_1 - 0.5r}{r} \tag{3.119}$$

where: r is a number of rows in the spherical image and c is a number of columns in the spherical image. The center of the image is a pixel with coordinates $\frac{c}{2}$, $\frac{r}{2}$. It lies on the intersection of the equator of the projection sphere and the prime meridian. Increasing v coordinate (vertical) goes towards the bottom of the image and moves the projected point in the south direction, thus decreasing longitude. Increasing u coordinate (horizontal) goes towards the left side of the image and moves the projected point in the west direction, thus increasing latitude. To find a great circle projection with the azimuth α and the inclination β for N points, the equation (3.120) needs to be minimized.

$$\arg\min_{\alpha,\beta}\sum_{k=1}^{N}\left\|\left(\arctan\left(\frac{-1}{\tan(\beta)(\sin(\alpha-\varphi_k))}\right) - \theta_k\right)\right\| \tag{3.120}$$

Equation (3.120) can be minimized using the least square method with automatic differentiation and the Levenberg-Marquardt algorithm [2]. Obtaining a normal vector of the great circle is straightforward once parameters α and β are found.

Extracting Plücker line from a point cloud

If provided with a point cloud, extracting Plücker line is essential for an automatic calibration of the camera's extrinsic parameters. Points which build a line feature can be easily classified using a closed form solution. The first step is the computation of the centroid of the points forming the line feature. It is $\bar{\boldsymbol{X}} \in \mathbb{R}^3$, as seen in equation (3.121).

$$\bar{\boldsymbol{X}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{X}_i \tag{3.121}$$

The covariance matrix $\boldsymbol{C}$ of the points is computed with equation (3.122).

$$\boldsymbol{C} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^{\intercal} \tag{3.122}$$

Matrix $\boldsymbol{C}$ is decomposed using SVD:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\intercal} \tag{3.123}$$

The line fitting problem is solved by taking a directional vector of the first column of $\boldsymbol{V}$. The first column of $\boldsymbol{V}$ shows the direction of the greatest change in $\boldsymbol{X}_i$. This vector is a directional component of a Plücker coordinate.

$$\boldsymbol{l} = \begin{bmatrix} \boldsymbol{V}(1,1) \\ \boldsymbol{V}(2,1) \\ \boldsymbol{V}(3,1) \end{bmatrix} \tag{3.124}$$

Moment $\boldsymbol{m}$ is orthogonal to a $\boldsymbol{l}$ and $\boldsymbol{x}$. Vector $\boldsymbol{x}$ is assembled from the centroid and points from the origin to the centroid of line.

$$\boldsymbol{m} = \boldsymbol{x} \times \boldsymbol{l} \tag{3.125}$$

where $\times$ is a vector cross product.

Extrinsic calibration of a spherical camera

Extrinsic calibration is parameterized using a Lie algebra $\mathfrak{se}(3)$ so that there are six parameters in total: $\boldsymbol{\rho} \in \mathbb{R}^3$ and $\boldsymbol{\sigma} \in \mathbb{R}^3$. Vector $\boldsymbol{\rho}$ represents a translational part of $\mathfrak{se}(3)$ and $\boldsymbol{\sigma}$ rotational part of $\mathfrak{se}(3)$. Note that $\boldsymbol{\rho}$ is trivial Lie algebra, so exponential mapping can be omitted. To obtain rotation $\boldsymbol{R}_{wc}$ and translation $\boldsymbol{t}_{wc}$ following mapping need to be performed:

$$\begin{bmatrix} \boldsymbol{R}_{wc} & \boldsymbol{t}_{wc} \end{bmatrix} = \begin{bmatrix} \exp(\boldsymbol{\sigma}^{\wedge}) & \boldsymbol{\rho} \end{bmatrix} \tag{3.126}$$

The residual for this problem is built from:

- a 3D line detected in a laser scan, described with a Plücker coordinate which lives in $\mathbb{R}^6$, $[\boldsymbol{m}, \boldsymbol{l}]^{\intercal}$,
- a 2D line detected in a spherical image, described with a pair of angles (azimuth α and inclination β) which is in $\mathbb{R}^2$.

Plücker coordinates $[\boldsymbol{m}, \boldsymbol{l}]^{\intercal}$ of a line detected in a laser scan are transformed to the currently estimated center of projection with formula (3.127). The formula (3.127) is obtained by substitution of (3.126) to (3.109).

$$\begin{bmatrix} \boldsymbol{m}_l \\ \boldsymbol{l}_l \end{bmatrix} = \begin{bmatrix} (\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal} & -(\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal}[\boldsymbol{\rho}]_{\times} \\ \boldsymbol{0}^{3x3} & (\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal} \end{bmatrix} \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{l} \end{bmatrix} \tag{3.127}$$

Therefore, the moment $\boldsymbol{m}_l$ can be taken from equation (3.127) resulting with equation (3.128).

$$\boldsymbol{m}_l = (\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal}(\boldsymbol{m} - [\boldsymbol{\rho}]_{\times}\boldsymbol{l}) \tag{3.128}$$

The great circle's normal vector $\boldsymbol{N}_g$ is built from α and β with equation (3.129). The equation (3.129) is result of taking $[a, b, c]$ from equation (3.111).

$$\boldsymbol{N}_g = \begin{bmatrix} \sin(\alpha)\sin(\beta) & -\cos(\alpha)\sin(\beta) & \cos(\beta) \end{bmatrix} \tag{3.129}$$

Summarizing:

- $\boldsymbol{m}_l$ is found from a line in a 3D space and the current orientation of the camera,
- $\boldsymbol{N}_g$ is obtained from azimuth α and inclination β with equation (3.129).

These vectors should be parallel, thus the optimization problem is defined by equation (3.130).

$$\underset{\boldsymbol{\rho},\boldsymbol{\sigma}}{\arg\min} \left\| \underbrace{\left[(\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal}(\boldsymbol{m} + [\boldsymbol{\rho}]_{\times}\boldsymbol{l})\right]}_{\boldsymbol{m}_l} \times \boldsymbol{N}_g \right\| \tag{3.130}$$

Equation (3.130) gives zero when $\boldsymbol{m}_l$ and $\boldsymbol{N}_g$ are parallel. That is a consequence of property of a length of a cross product of two vectors. The cross product length is proportional to sine of angle between those two vectors. Requesting the length to be zero in observation equation (3.130), the parameters $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ are found which makes $\boldsymbol{m}_l$ and $\boldsymbol{N}_g$ parallel requesting to angle between $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ to be 0 degree or 180 degrees.

3.5.2 Line to line in rectilinear image

The method introduced in the previous sections can be adopted for rectilinear cameras. Extrinsic calibration is parameterized using a Lie algebra $\mathfrak{se}(3)$ so that there are six parameters in total: $\boldsymbol{\rho} \in \mathbb{R}^3$ and $\boldsymbol{\sigma} \in \mathbb{R}^3$. Vector $\boldsymbol{\rho}$ represents a translational part of $SE(3)$ and $\boldsymbol{\sigma}$ rotational part of $SE(3)$. To obtain rotation $\boldsymbol{R}_{wc}$ and translation $\boldsymbol{t}_{wc}$ the mapping (3.126) need to be performed. A line in a 3D space is described by Plücker coordinates. The moment vector $\boldsymbol{m}$ and direction vector $\boldsymbol{l}$ can be retrieved from pointcloud using equations (3.121) to (3.125). The vector $[\boldsymbol{m}_l, \boldsymbol{l}_l]^{\intercal}$ can be obtained transforming $[\boldsymbol{m}, \boldsymbol{l}]^{\intercal}$ from global coordinate frame to camera's local coordinate frame using equation (3.109). Taking a normalized vector $\boldsymbol{n}_l = \boldsymbol{m}_l / \|\boldsymbol{m}_l\|$ from the local Plücker coordinate gives a normal vector for a plane which goes through the observed line, and the projection center of the camera in the camera's coordinate frame. All major vectors and geometries are shown in drawing 3.15.

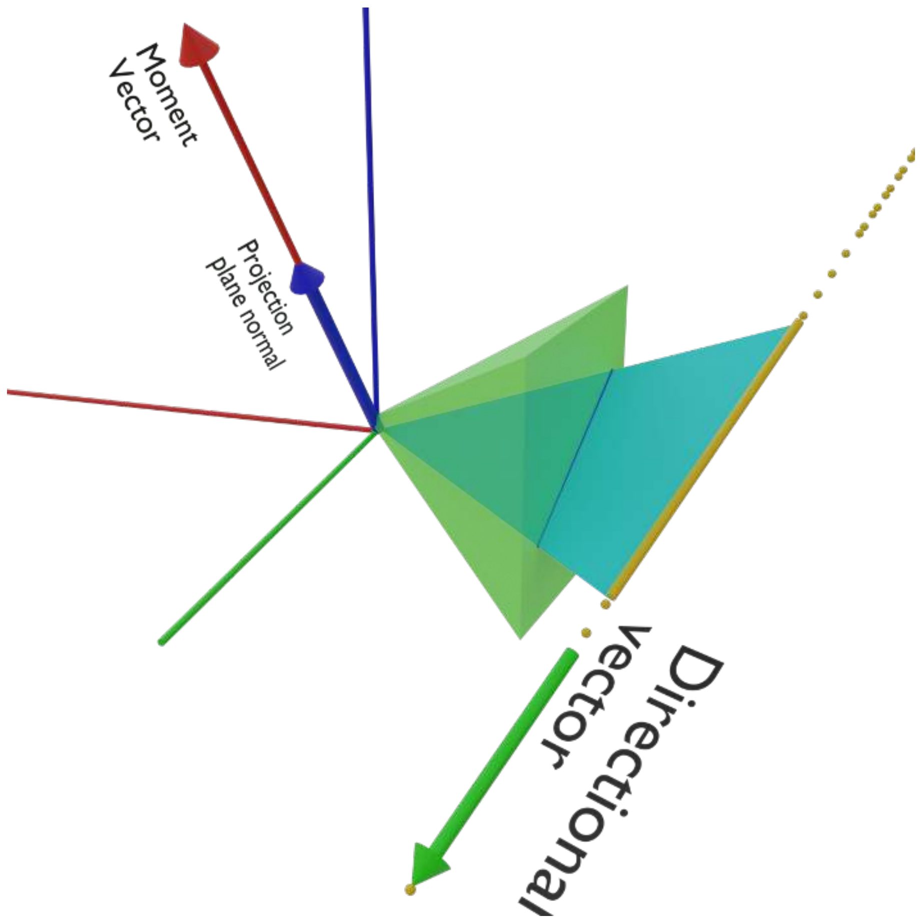


FIGURE 3.15: The line segment in a 3D space is marked in yellow. There are two vectors that create the Plücker coordinate: the moment is marked in red and the directional vector is marked in green. The projection plane spans on the center of the projection (marked in green) and the observed line (yellow). This plane contains an image of the line (marked in blue). A normal vector to projection plane is marked in blue.

The line in the image is built by a set of 2D points with coordinates: $p_{x,1}, p_{y,1}$,$p_{x,2}, p_{y,2}$ ... $p_{x,n}, p_{y,n}$. This line is presented as a vector $\boldsymbol{v}_m$. $\boldsymbol{v}_m$ is normal to a plane which spans on the line on the projection plane and the projection center. The projection of the plane to the image space is given with equation (3.131).

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \boldsymbol{K}\boldsymbol{v}_m \tag{3.131}$$

Matrix $\boldsymbol{K}$ is a projection matrix given with equation (3.132) [7].

$$\boldsymbol{K} = \det(\boldsymbol{P})\boldsymbol{P}^{-\intercal} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ -c_x f & -c_y f & f^2 \end{bmatrix} \tag{3.132}$$

where : $\boldsymbol{P}$ is camera projection matrix. $\boldsymbol{K}$ and $\boldsymbol{P}$ contains the following camera parameters: f_x, f_y - optical lengths, c_x, c_y - principal point. The proof for equation (3.132) can be found [7]. To find a $\boldsymbol{v}_m$ for given 2D points, the optimization problem is constructed as equation (3.133).

$$\arg\min_{\boldsymbol{v_m}} \sum_{n=1}^{N} \left\| (\boldsymbol{K}\boldsymbol{v}_m) \begin{bmatrix} p_{x,n} \\ p_{y,n} \\ 1 \end{bmatrix} \right\| \tag{3.133}$$

Summarizing :

- $\boldsymbol{v}_m$ is found from a line in the image from a set of points,
- $\boldsymbol{m}_l$ is found from a line in a 3D space and the current $SE(3)$ pose of the camera.

These vectors should be identical regarding the direction, thus the optimization problem is defined by equation (3.134).

$$\arg\min_{\boldsymbol{\rho},\boldsymbol{\sigma}} \left\| \underbrace{\left[(\exp(\boldsymbol{\sigma}^{\wedge}))^{\intercal}(\boldsymbol{m} + [\boldsymbol{\rho}]_{\times}\boldsymbol{l})\right]}_{\boldsymbol{m}_l} \times \boldsymbol{v}_m \right\| \tag{3.134}$$

Equation (3.134) gives zero when $\boldsymbol{m}_l$ and $\boldsymbol{v}_m$ are parallel.

Finally, those observation equations are closing elaborated methodology. In the next two chapters, the presented methodology will be verified against multiple real-world challenges. Developed solutions and algorithms in chapters Experimental validation and Robotic applications are direct implementation of the methodology. A prominent example of such implementation is rotation representation that utilizes Lie algebra, or LM algorithm that is mainly used for solving calibrations problems formulated as IRLS optimization problems.

Chapter 4

Experimental validation

This chapter presents multiple usage scenarios for methods and tools presented in the previous chapter. Calibration problems shown in the next few examples are real-world systems which required automatization to operate. Solving these problems required designing or adopting new techniques. In section 4.1 the calibration problem was solved by designing factor-graph SLAM with extra hidden variables which represented a geometrical configuration of the system. Other systems (4.2 and 4.3) required incorporating equations of a reflected ray into point to point ICP. Section 4.4 demonstrates the approach to calibrating the mechanical imperfections of the system with a simple calibration pattern utilizing design features of the sensor. The system described in section 4.5 presents a challenges accompanying extrinsic calibration for the sensors that do not have any overlaps in its FOV. Section 4.6 contains a description of a calibration of a spherical camera.

4.1 Calibration of a high-volume mobile 3D scanner

The first calibrated system uses two RGBD (Intel RealSense L515) sensors which are rotated using an arm. One of the scanning heads is shown in figure 4.5. The basic design of the system is shown in schematic 4.1 and the system scanning calibration pattern is shown in figure 4.6. The scanning heads are attached to linear actuators. Those actuators are attached to the rotating arm. The whole assembly is rotated around the scanned volume. The incremental encoder attached to the arm measures the current angle of the revolution. The arm is motorized with a DC servo actuator. The sensors' rig can be rotated around a horizontal axis which goes through the center of the scanned volume. The sensor can be moved parallelly (up and down) by a pair of motorized actuators. The rig has been designed and assembled using CAD software and it has had known dimensions, but it can be assembled in several different ways. Manual calibration or even pre-calibration is not possible due to a complicated mechanical structure.

The RGBD scanner head (Intel RealSense L515) is composed of a visible light RGB camera and a solid-state LiDAR camera. The manufacturer provides the following calibration of these sensors:

- focal length and principal point for depth sensor $f_{dx}, f_{dy}, c_{dx}, c_{dy}$,
- rectification coefficients for Brown-Conrady (so called 'plumb-bob') distortion model K_1, K_2, K_3, K_4, K_5 [57],
- focal length and principal point for RGB sensor $f_{cx}, f_{cy}, c_{cx}, c_{cy}$,

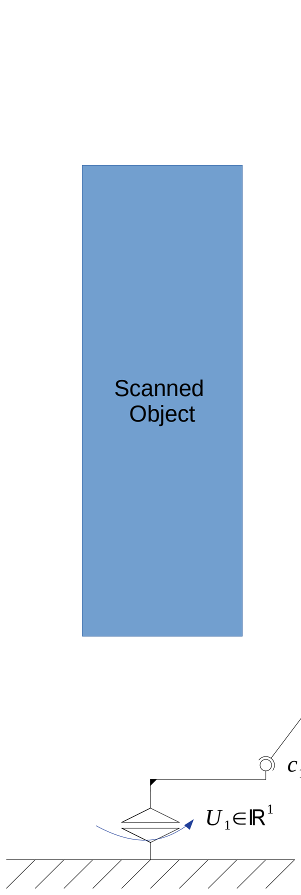


FIGURE 4.1: Mechanical structure of the high-volume mobile 3d scanner. Actuated revolution joint u_1, actuated prismatic joints: u_2, u_3. Revolution joints with hidden configuration: c_1,c_2,c_3,c_4.

- extrinsic calibration of depth optical frame and RGB optical frame $\boldsymbol{T}_{cd}$.

This information is enough to build a colored point cloud from the measurements without any extra calibration [66]. A chain of these operations can be used to yield a 3D point in color. First, an in-depth image u_d, v_d with known depth z_d is projected in a 3D space:

$$\begin{bmatrix} X_d^g \\ Y_d^g \\ Z_d^g \end{bmatrix} = \begin{bmatrix} z_d(u_d - c_{dx})/f_{dx} \\ z_d(v_d - c_{dy})/f_{dy} \\ z_d \end{bmatrix} \tag{4.1}$$

The obtained point $[X_d^g, Y_d^g, Z_d^g]^\intercal$ is in the depth's camera local coordinate system. The point needs to be transformed to a color camera coordinate system:

$$\begin{bmatrix} X_c^g \\ Y_c^g \\ Z_c^g \\ 1 \end{bmatrix} = \boldsymbol{T}_{cd} \begin{bmatrix} X_d^g \\ Y_d^g \\ Z_d^g \\ 1 \end{bmatrix} \tag{4.2}$$

Finally, $[X_c^g, Y_c^g, Z_c^g]^\intercal$ is projected onto the color camera coordinates $[u_c, v_c, 1]$.

$$\begin{bmatrix} su_c \\ sv_c \\ s \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 & c_{cx} \\ 0 & f_{cy} & c_{cy} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_d^g \\ Y_d^g \\ Z_d^g \end{bmatrix} \tag{4.3}$$

Color camera coordinates $[u_c, v_c, 1]$ can be taken to find a color of point taking a pixel color in a rectified image. The calibration procedure needs to give an exact geometrical configuration of the system. Fiducial makers are used to solve this problem. The exact type of markers and detection method, as well as implementation and processing details are explained in [105] and [46]. Fiducial markers enable the system to accurately find markers' boundaries and, if intrinsic calibration of the camera is known, to find $SE(3)$ transform of the marker in a 3D space. In this approach, only the identification of fiducials in the camera image is used. Fiducials detection solution allows for an accurate and robust detection of the corners of a marker and its identification number. The marker and its corners create a polygon. The processing of fiducial markers is shown in figure 4.2.

Every point from the point cloud, provided by a LiDAR camera, is tested if it lays in the polygon. If the point lays in the polygon, it is marked with the identification number of the marker detected in the RGB image frame. In the figure 4.2 depth pixels which belong to different markers are marked with different colors. The unique identification numbers of the markers are marked in various colors.

Geometrical information about the location is robust enough to track given 3D correspondences in multiple 3D scan poses. Geometrically, the collection of the point which share the same identification number is collapsed to a single centroid, resulting in a single 3D point. It is demonstrated in figure 4.3.

A corrupted observation may occur at this point, e.g. due to occlusion or extreme observation angle. It can be eliminated by performing SVD and analyzing diagonal matrix $\boldsymbol{\Sigma}$. One application of SVD (e.g, in PCA) is change of basis of the dataset. In case of pointcloud, it allows to recover new 3D coordinate system that axis spans into direction of most significant variance. A correctly observed marker should not be significantly distorted by perspective projection. The covariance matrix of points which share the same identification number is computed. SVD decomposes the covariance matrix into two orthogonal matrices and one diagonal. Two first elements of the diagonal of $\mathbf{\Sigma}$ are limited to be in some tuned vicinity of 1.0. This means that only the markers which have similar width and height are kept. The procedure is done for multiple capture images resulting in sets of 3D points with an identification number. This method is considered a part of the proposed methodology. Thus, it automates the calibration of the multi sensor mapping system.

Modeling a mechanical design as a factor graph

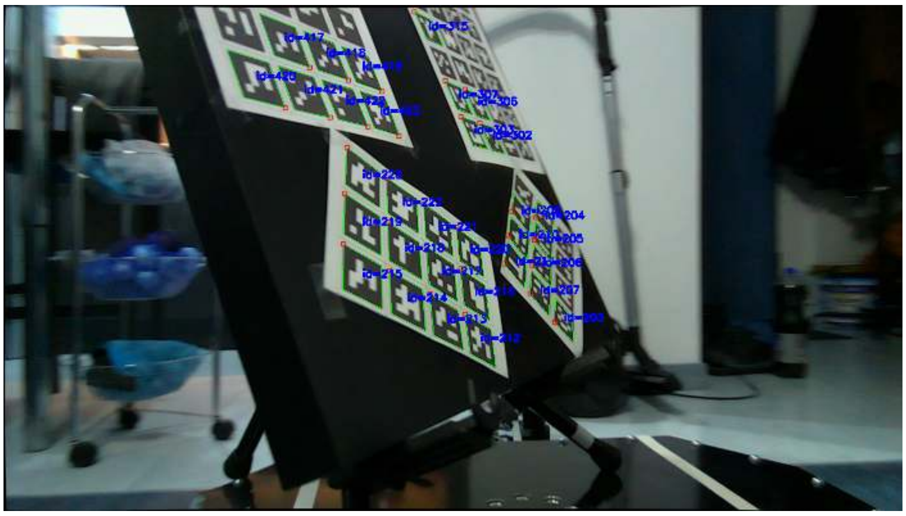
This system can be represented with a mechanical diagram shown in figure 4.1. One of the two scanning heads is shown in figure 4.5. The complete calibrated system is shown in figure 4.6. The system consists of four rotation joints which are assembled by the user and fixed in place, two prismatic joints which are actuated, and one rotation joint which is actuated. The assembled joints are fixed and their configuration is unknown. The electric motors move actuated joints and their current position is measured with sufficient precision. Thus, it is negligible in the optimization problem. That means every assembled joint contributes six degrees of freedom to the optimization problem. The system has twenty-four degrees of freedom in total.

To recover the current state of the system ($SE(3)$ configuration of c_1,c_2,c_3,c_4) after the assembly of the device, some constraints and relations need to be utilized. The problem partially fulfills the SLAM definition; the map is unknown and the trajectory is partially known. It differs from the classical SLAM, as the trajectory is dependent only on the calibration of the system. Said calibration has only twenty four degrees of freedom, while a trajectory in a small SLAM problem can have a few hundred degree of freedom. Such system can be represented as a factor graph, as shown in figure 4.4. It results in a clear design of the optimization problem. Factors $f_0(x_0, \theta, U)$, $f_1(x_1, \theta, U)$, $f_2(x_2, \theta, U)$, $f_3(x_3, \theta, U)$ show that poses x_0,x_1,x_2,x_3 are dependent on a random variable θ and controls U. Note that, controls U, which are positions reported by actuated joints, are not optimized parameters. Here the system differs from a typical SLAM or self-calibrated graph-SLAM [111]. The crucial difference is that poses x_0,x_1,x_2,x_3 are not linked by any factor with each other, what is typical in SLAM. Factors $f_4(x_0, l_0)$,$f_5(x_1, l_0)$,$f_6(x_2, l_0)$,$f_7(x_1, l_1)$,$f_8(x_2, l_1)$,$f_9(x_3, l_1)$ are representing observations. It is essential to point out that poses x_0,x_1,x_2,x_3 are not estimated in the final problem of the likelihood maximization. They are dependent on one random variable θ and the reported position of actuated joints. The position is treated as a constant parameter in the optimization problem. The calibration method maximizes the likelihood related to the projection of landmarks visible from multiple poses. The problem is optimized with Levenberg-Marquardt non-linear least square method [80]. The method converges on a sample calibration field 4.7. Final cost Root Mean Square (RMS) (per observation) is 5.01 mm. The evaluation of the performance of this method is quantitative. Some quantitative reasoning shows that optimized parameters give similar results for the multiple datasets and captures (average error in millimeters): The table 4.1 shows an average distance of the observation in millimeters. The rows represent the results on calibration data-sets and test against other datasets. It is visible that the algorithm yields

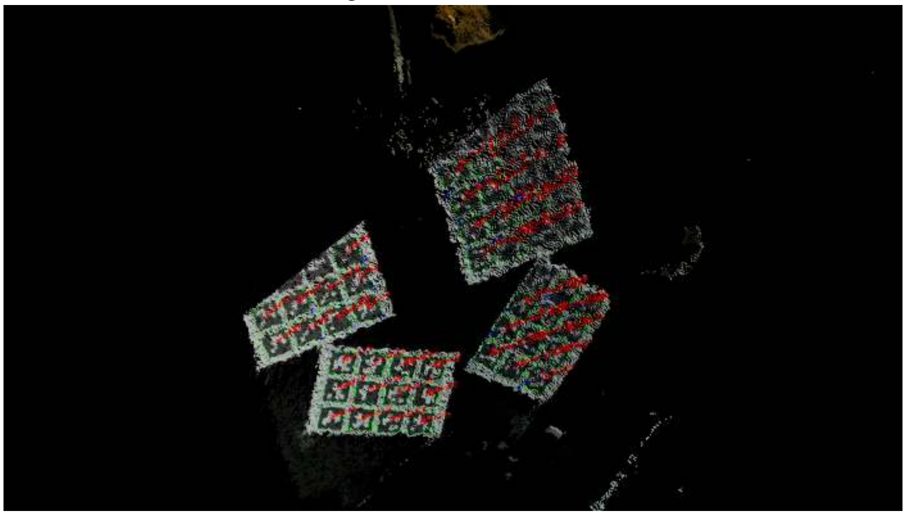
TABLE 4.1: Comparison of cumulative error on four calibration datasets. For every row, one dataset was used to obtain calibration and rest was used to assess accuracy.

calib./test.	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Dataset 1	5.01 mm	7.34 mm	7.46 mm	5.58 mm
Dataset 2	7.93 mm	5.69 mm	10.15 mm	9.09 mm
Dataset 3	7.63 mm	9.58 mm	6.57 mm	6.69 mm
Dataset 4	5.80 mm	7.89 mm	6.90 mm	5.02 mm

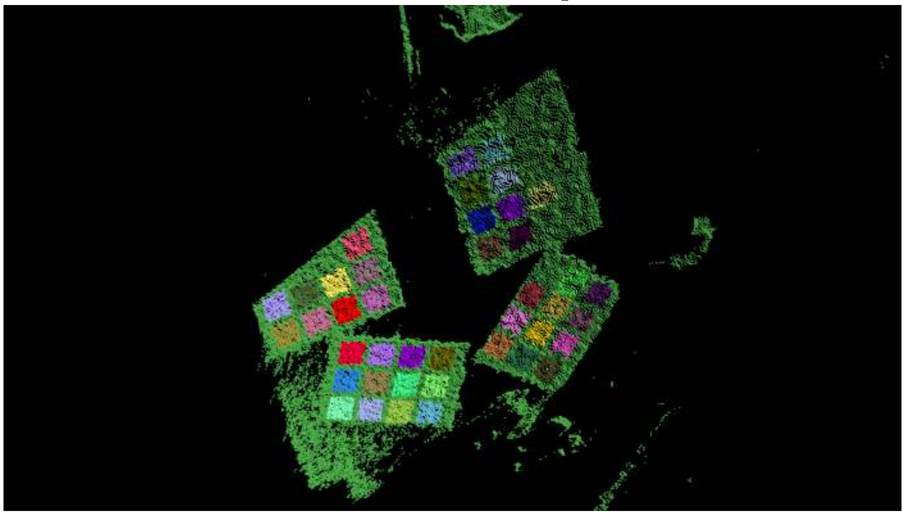
the lowest error in tests performed on the data-set used for calibration (the diagonal of the array contains the lowest value). Datasets 2 and 3 perform poorly and yield the worst calibration (tested against datasets 1 and 2). Calibration provided by Datasets 1 or 2 has a significantly smaller score on the tests against datasets 2 and 3. It is so because captures 2 and 3 were done with greater rotation speed. As a result, the number of captured fiducials was smaller due to the motion blur. The method converges well both on real and synthetic datasets obtained by Blender Cycles raytracing engine (described in section 3.3.6). Synthetic data-set converges to negligible error. It is shown in figure 4.7.



(A) RGB image with with detected markers.

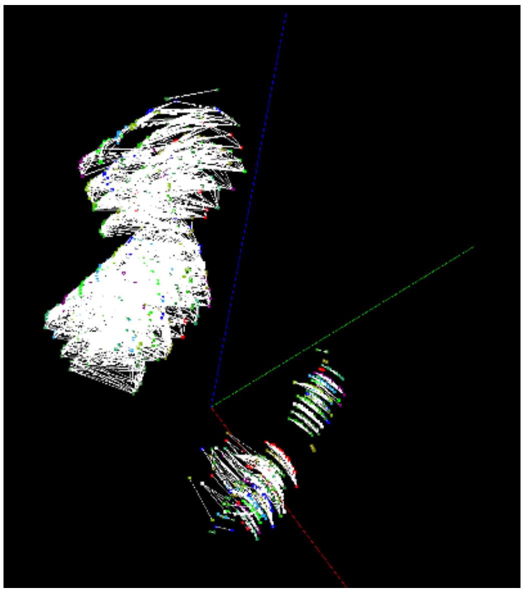


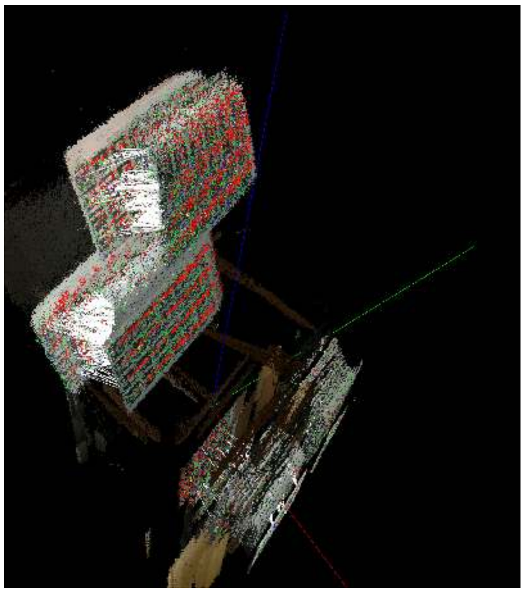
(B) Fiducial marker in RGBD pointcloud.

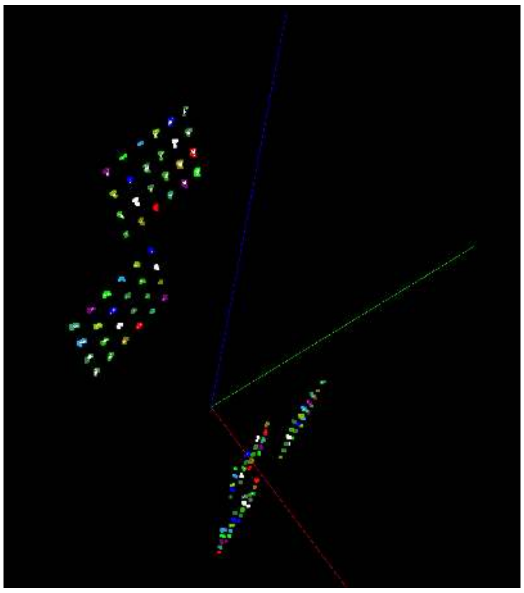


(C) Point-cloud classified with underlying AR marker code.

FIGURE 4.2: Processing of fiducial markers.


(A) Centroids of markers before calibration.


(B) RGBD, before calibration .


(C) Centroids of markers after calibration.

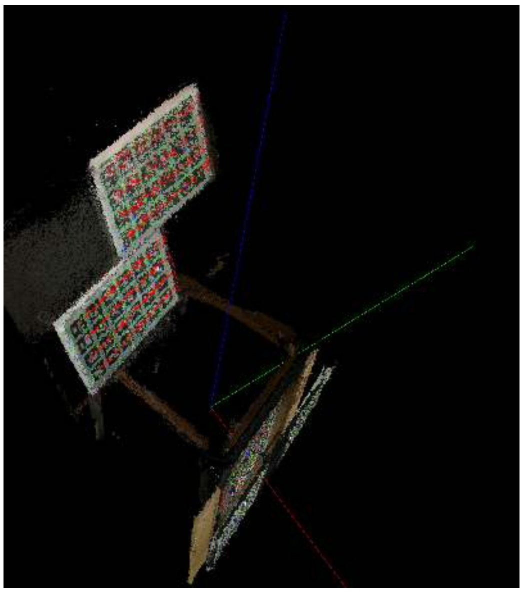

(D) RGBD, after calibration.

FIGURE 4.3: Detected fiducials before and after automatic calibration.

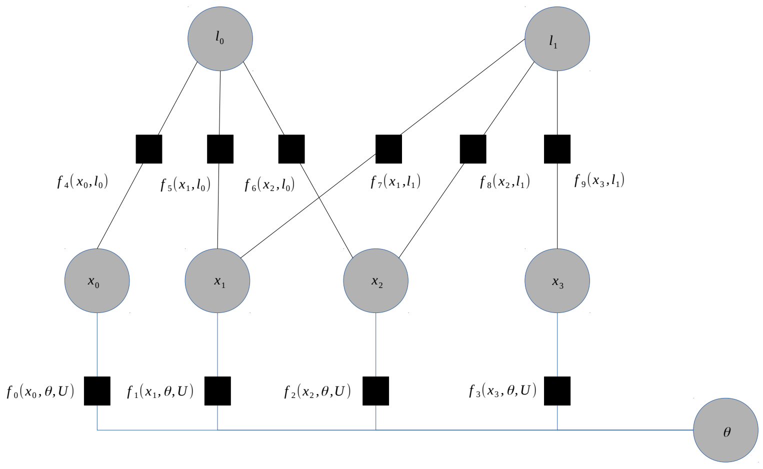


FIGURE 4.4: Factor graph modeling the calibrated system. U - controls (configuration of actuated joints). x_0, x_1, x_2, x_4 - poses of calibrated camera. $l1, l2$ - observed centroids. $\theta = \{c_1, c_2, c_3, c_4\}$ - calibrated parameters (configuration of assembled joints).

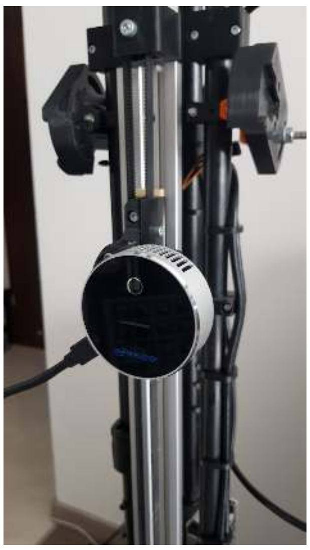


FIGURE 4.5: One of the two scanning heads. There is a visible guiding system with a timing belt in the back. In the front, there is an RGBD sensor with an RGB camera and LiDAR lens.

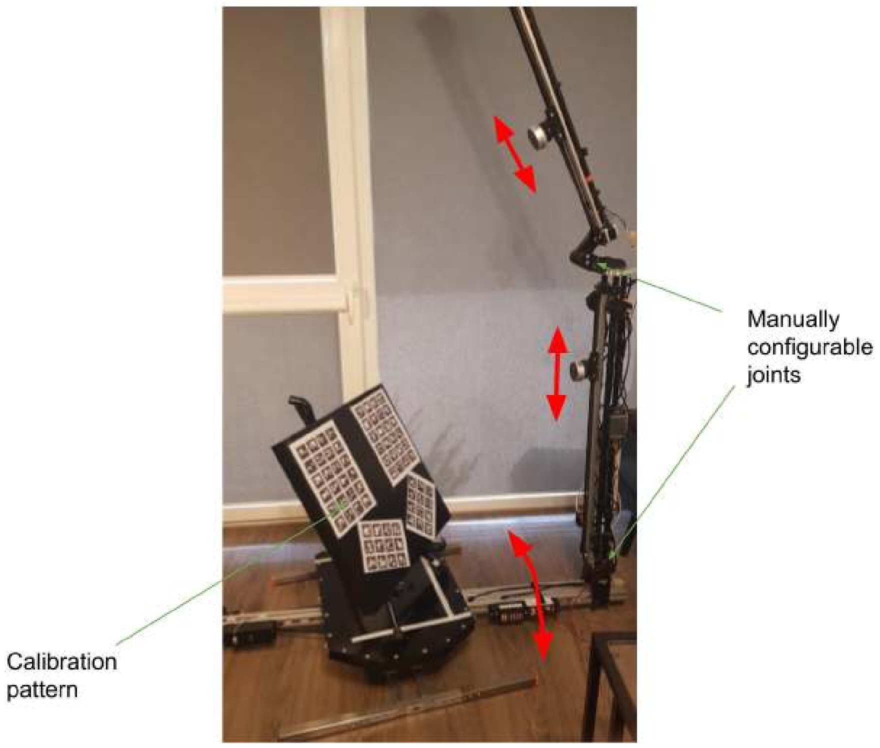


FIGURE 4.6: The complete calibrated system. The calibration field is a combination of a large number of AR markers. Mechanically, the system consists of two guided rails with a timing belt linear transport for two scanning heads marked in red. The arm on the bottom is rotated by a DC-servo motor (curved red arrow). The system can be assembled in multiple ways to achieve the best coverage of the scanned volume.

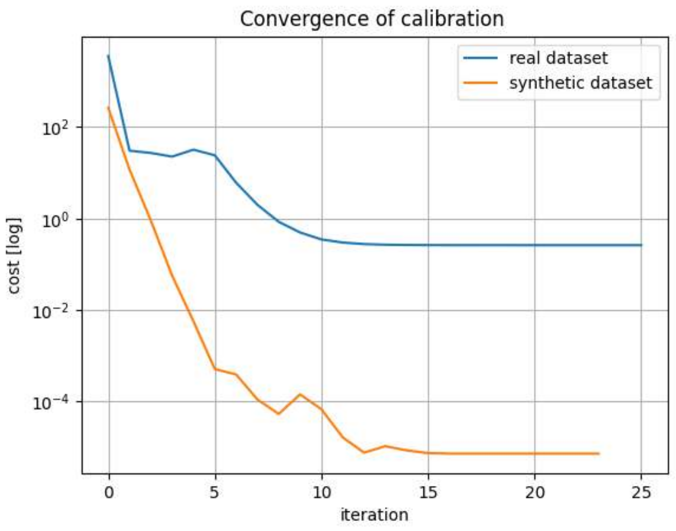


FIGURE 4.7: Converging cost with iterations of the Levenberg-Marquadt algorithm for synthetic and real dataset. The cost is total distance. 14710 observations were used in the real dataset. 1074 observations were used in the synthetic one.

4.2 Calibration of multi planar reflectors

This system includes a Livox Mid-40 LiDAR laser scanner [79] (figure 4.8). This LiDAR sensor is similar to a Velodyne VLP-16 [130], but it has a completely different field of view. The Livox Mid-40 LiDAR has an initially conical field of view with the apex in the optical center and the apex angle equal to 38.4 degrees. Conversely, the Velodyne VLP-16 has a cylindrical field of view. The Livox Mid-40's field of view has a smaller coverage of the scene with greater angular resolution. The goal is to reshape the field of view of the Livox Mid-40 to be similar to other classical rotating LiDARs. Therefore, the calibration process is designed to cope with this new sensor. This device splits the Livox Mid-40's field of view into six separate beams, which allows for a broader coverage of the scene. It is desired for slow-moving robotic agents in known environments for localization purpose.

Mechanical design

The Livox LiDAR is oriented with its aperture pointing up. A pyramidal structure with mirrors is installed on top of the Livox LiDAR. Three pillars support the structure. They are assembled to a face plate which is screwed to the front piece of the Livox LiDAR, as seen in the simplified construction drawing in figure 4.8.

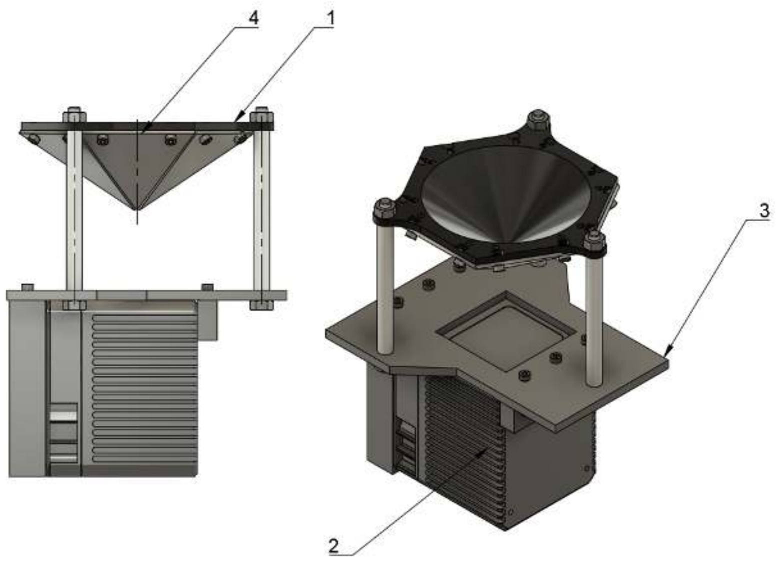


FIGURE 4.8: 3D CAD model of assembly. 1- hexagonal pyramid; 2-Livox LiDAR; 3- face plate; 4- one of six mirrors.

The field of view after the modification consists of six segments with the following properties:

- the field of view (vertical) spreads from -12° to 9.4° (Fig. 4.9),
- the field of view (horizontal) has six segments orientated radially with angle up to 18.7° (Fig. 4.10).

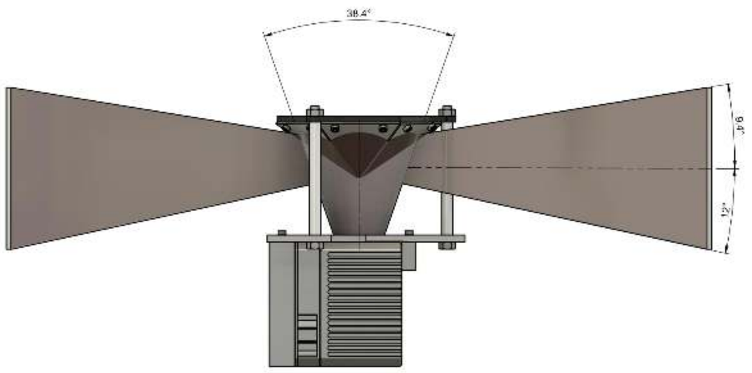


FIGURE 4.9: Reshaped field of view. A plot of vertical situation.

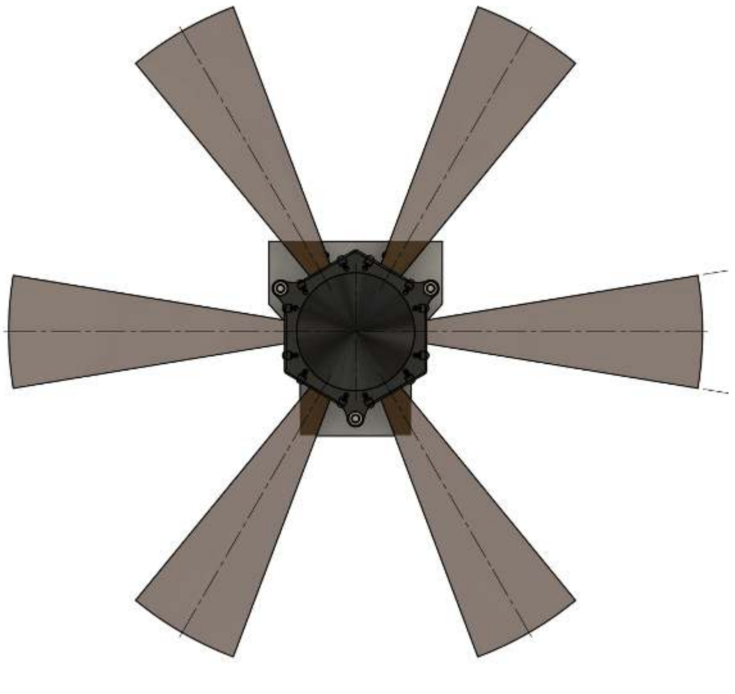
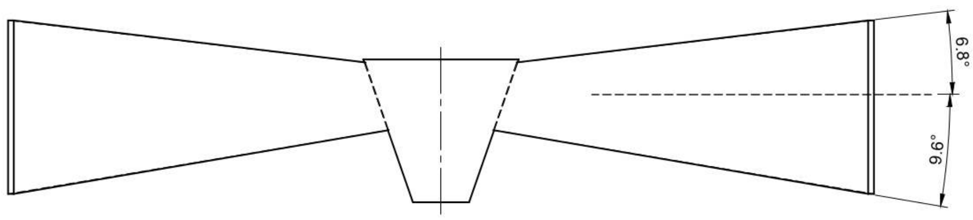
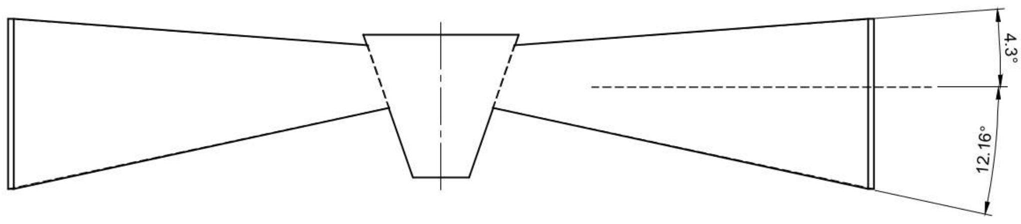


FIGURE 4.10: Reshaped field of view, plot of horizontal situation.

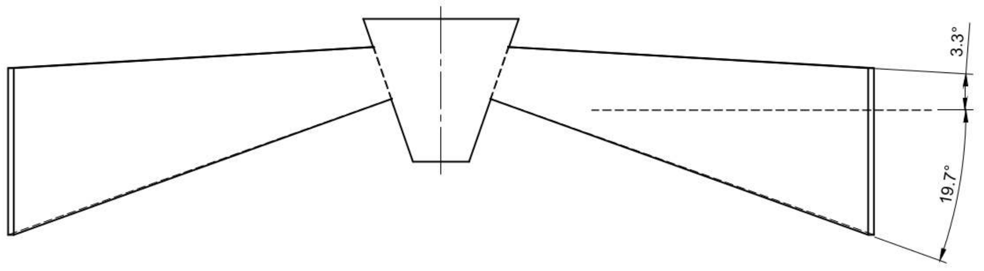
The shape of the resulting field of view in the vertical plane depends mainly on the angle between the central axis and the mirror surface (Fig. 4.11). The horizontal field of view depends on the number of mirrors. A larger number of mirrors would result in the larger number of radial segments. Unfortunately, every edge introduces artifacts and causes some rays to point into the spacing between the mirrors. A hexagonal pyramid (element 1 in figure 4.8) was 3D printed out of Polylactic Acid (PLA) using a standard Fused Deposition Modelling (FDM) printer. This structure has multiple embedded nuts used to screw in six triangular mirrors. Mirrors are made out of a reflective Poly Methyl Methacrylate (PMMA) sheet that was cut using Computerized Numerical Control (CNC). Each mirror is screwed to the hexagonal pyramid.



(A) Mirror angles 50°.



(B) Mirror angles 52.5°.



(C) Mirror angles 60°.

FIGURE 4.11: The impact of the mirror angle on the changing vertical field of view.

The pyramid is screwed to the three pillars supporting the structure. The pillars are rods with a thread on each end. The construction is sturdy, but there is no way to ensure the parallelism of the optical axis of the Livox Mid-40 and the axis of the hexagonal pyramid. The manufacturer does not provide any dimensional or shape tolerance on the optical center of the Livox Mid-40. As a result, the center of the pyramid is shifted by an unknown value with respect to the optical center. This fact dictates the necessity for calibration.

Geometry of a reflected ray

The Livox Mid-40 emits a bundle of rays in a conical pattern. Technically, it emits one ray projected in the desired direction by two rotating prisms. The construction details of this class of LiDARs can be found in [77]. The rays are reflected by a mirror with the following properties:

- a plane equation (4.4), where $\mathbf{V}^{pl}$ is plane's normal given (4.5) and is unit-length (4.6)
- a three 2D corners which lay on a plane, which limits the dimension of the mirror.

$$ax + by + cy + d = 0 \tag{4.4}$$

$$\mathbf{V}^{pl} = \begin{bmatrix} a & b & c \end{bmatrix} \tag{4.5}$$

$$\left\| \mathbf{V}^{pl} \right\| = 1 \tag{4.6}$$

First, the intersection of the ray and the mirror plane is found. The ray starts at $[0, 0, 0]$ and points in the direction of unit vector $\mathbf{r}^b$. This intersection point is a linear scaling l^{int} of vector $\mathbf{r}^b$, equation (4.7).

$$\boldsymbol{P}^{int} = l^{int}\mathbf{r}^b \tag{4.7}$$

This intersection point satisfies the equation (4.8).

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} l^{int}\boldsymbol{r}^b \\ 1 \end{bmatrix} = 0 \tag{4.8}$$

The equitation (4.8) needs to be solved for l^{int}. Having the exact value of l^{int} for given ray $\mathbf{r}^b$ the intersection point $\boldsymbol{P}^{int}$ can be obtained with equation (4.7). If the intersection point p lays in the boundary of the mirror, it is projected on the mirror plane and the 2D point in the polygon test is performed. The mirror is a convex polygon, so the winding number algorithm is used [61]. In other words, the algorithm computes the angle subtended by each side of the convex polygon with the tested point $\boldsymbol{P}^{int}$. The point lies outside the polygon if the sum of all angles adds up to 2π. If a given ray is reflected by a given mirror, a direction of reflected ray $\boldsymbol{r}^d$ is given with equation (4.9). A geometrical interpretation of the equation (4.9) is shown in the figure 4.12.

$$\mathbf{r}^d = -2(\mathbf{b}^d \cdot \mathbf{V}^{pl})\mathbf{V}^{pl} - \mathbf{b}^d \tag{4.9}$$

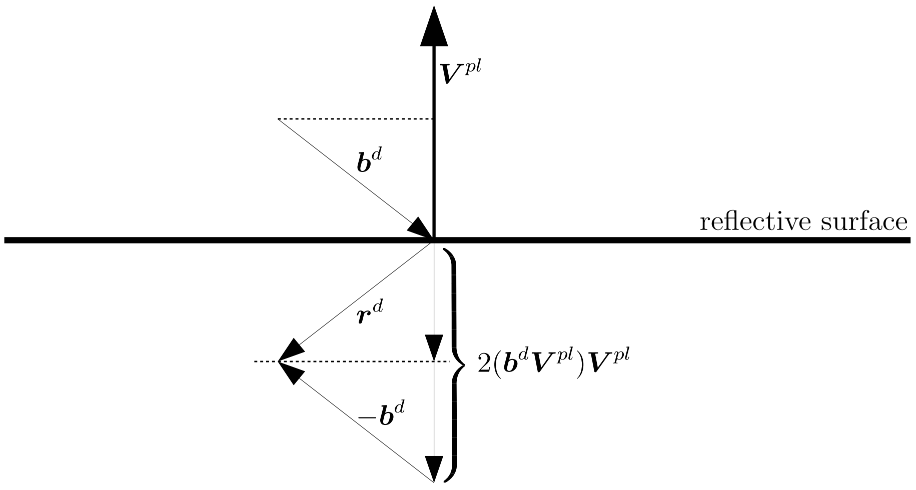


FIGURE 4.12: Geomerical interpretation of equation (4.9). $\mathbf{b}^d$ is a unit vector representing the direction of the incident beam. $\mathbf{r}^d$ is a unit vector representing the direction of the reflected beam. $\mathbf{V}^{pl}$ represent a mirror's normal vector. The dot product of $\mathbf{b}^d \cdot \mathbf{V}^{pl}$ is the length of the projection of $\mathbf{b}^d$ on $\mathbf{V}^{pl}$. Finally, the sum of vectors $-2(\mathbf{b}^d \cdot \mathbf{V}^{pl})\mathbf{V}^{pl}$ and $-\mathbf{b}^d$ results in $\mathbf{r}^d$ vector.

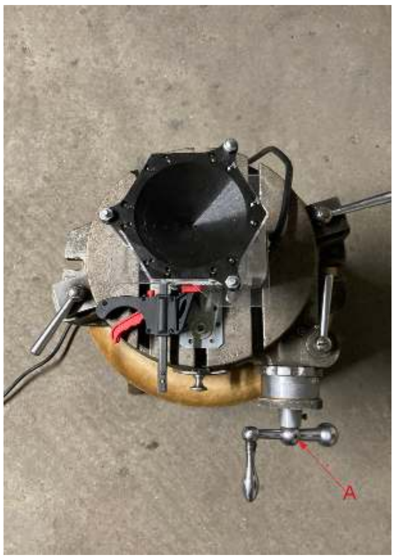
To get a point in a 3D space after reflection ($\mathbf{P}^r$) having $\mathbf{P}^l \in \mathbb{R}^3$ in local coordinate system one must employ equation (4.10):

$$\mathbf{P}^r = -(\mathbf{P}^{int} + \mathbf{r}^d(l^p - l^{int})) \tag{4.10}$$

where: $l^p = \left\|\mathbf{P}^l\right\|$ is the length of the beam from its origin to $\mathbf{P}^l$, $l^{int} = \left\|\mathbf{P}^{int}\right\|$ is the distance to the intersection from the origin.

Initially, the geometrical calibration was derived from a CAD design. The CAD design used for manufacturing is exported to Blender, where a simplified mesh is created. The simplified mesh consists of separate triangles, each of which represents a mirror. The initial configuration is saved as a Polygon File Format (PLY) file. The PLY file is loaded, and each mirror and its boundaries are found. The parsing of PLY files is straightforward. A plane is fit for each triangle. The centroid and the covariance matrix are computed. Next, the latter is decomposed using SVD. The third column of $\boldsymbol{U}$ is taken as a normal vector. d coefficient is obtained by the substitution of centroid coordinates to the equation (4.4) and solving for d. The $SE(3)$ transform is found by treating $\boldsymbol{U}$ as a rotation matrix and the centroid as a translation member of $SE(3)$. The inverse of this transform is used to project a ray-mirror intersection point back onto the mirror plane and to perform a point-in-polygon test. Unfortunately, the CAD-derived calibration is rather poor. To improve this calibration, the optimized plane coordinates need to be found for each of the mirrors. Poor performance of CAD-derived calibration is a result of:

- an imperfect alignment of the pyramid axis and the Livox Mid-40's axis,
- the thickness of the mirror PMMA surface,
- an imperfection in manufacturing.



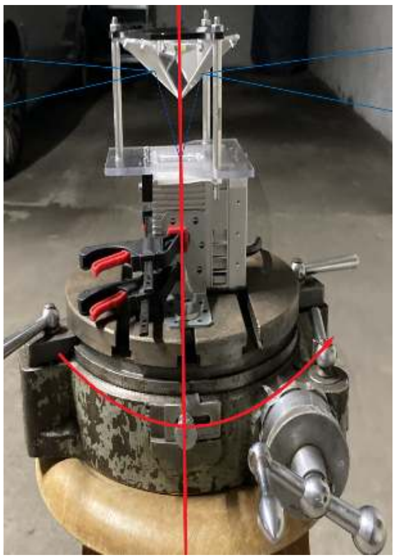


FIGURE 4.13: The experimental prototype mounted onto a precise rotating table during the data acquisition procedure. 'A' marks the axis for changing the rotation angle.

Calibration procedure

The calibration is performed in a stop-scan manner. The Livox Mid-40 with FOV changing mirrors is mounted on the top of a turntable. The turntable allows for a precise rotation angle with a resolution of 0.5°. In addition, the sensor's rotation enables multiple overlapping of data reflected with different mirrors. The calibration data can be captured in multiple stations. Thirty-six static measurements for each measurement station are captured by rotating the table by 10 degrees (figure 4.13). Each static measurement consists of three seconds of the recorded Livox Mid-40 data. The center of the rotated system draws a circle. The circle is small if the sensor's axis is aligned with the rotation axis. In the calibration procedure a reflected point $\mathbf{r}^d$ is transformed by a chain of $SE(3)$ transform to a global coordinate frame.

Going from left to right from the sensor's frame to the world's frame, equation (4.11):

- $\begin{bmatrix} \mathbf{R}^{lt} & \mathbf{t}^{lt} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \in SE(3)$ - *laser to turntable* represents the imperfect placement of the Livox LiDAR on the turning table. In a perfect scenario there should be identity as the calibration algorithm optimizes this transformation. This parameter is stationary for the experiment as the Livox LiDAR is not moving against the turning tables attachment point.

- $\begin{bmatrix} \mathbf{R}^{tp} & \mathbf{t}^{tp} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \in SE(3)$- *turntable to pose*, this transformation, which is a simple rotation around 'X', represents an angle set on the turning table. This parameter is not optimized by calibration algorithm.

- $\begin{bmatrix} \mathbf{R}^{pg} & \mathbf{t}^{pg} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \in SE(3)$ - *pose to global*, this transformation transforms a measurement station frame to the global world's frame.

All transformations together transform point $\mathbf{P}_a^r$ to a point in global coordinate frame $\mathbf{P}_a^g$:

$$\begin{bmatrix} \mathbf{P}_a^g \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{pg} & \mathbf{t}^{pg} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}^{tp} & \mathbf{t}^{tp} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}^{lt} & \mathbf{t}^{lt} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_a^r \\ 1 \end{bmatrix} \tag{4.11}$$

The optimized cost function for a corresponding of points $\mathbf{P}_a^g$ and other point in global coordinate system $\mathbf{P}_b^g$

$$\mathbf{c}_{ab} = \mathbf{P}_a^g - \mathbf{P}_g^b \tag{4.12}$$

For each of these parameters, a Lie algebra $\mathfrak{se}(3)$ was used to parametrize the pose. In the optimization problem the mirror was represented as a 4D vector. The coefficients of the plane were optimized. Since the data was treated as not structured, an exhaustive k-nearest neighborhood search had to be performed in search for correspondences. Kd-tree algorithm was used (Point Cloud Library (PCL) in the implementation [108]).

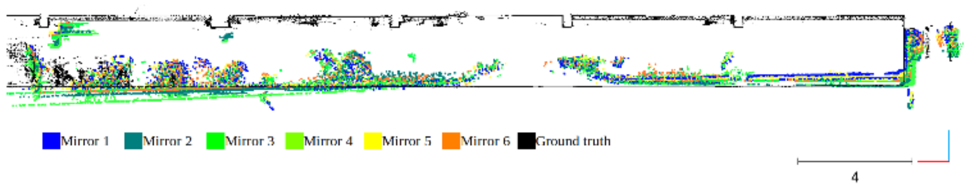
Only one station is used in the example below. The best available calibration is taken to construct a pointcloud from the raw data in the global coordinate system. In the next step, a Kd-tree is built on top of the pointcloud. Using the Kd-tree, the algorithm searches for pairs of the nearest neighborhood points with each individual point in a pair reflected by a different mirror. For every found nearest neighborhood point pair $(\mathbf{P}_a^g, \mathbf{P}_b^g)$, a residual $\boldsymbol{c}_{ab} \in \mathbb{R}^3$ is created. The point $\boldsymbol{P}_a^g$ was observed:

- with the measurement $\mathbf{b}_a^d$ (direction of ray),
- with distance d_a,
- reflected by the mirror γ,
- while the laser scanner was at angle β.

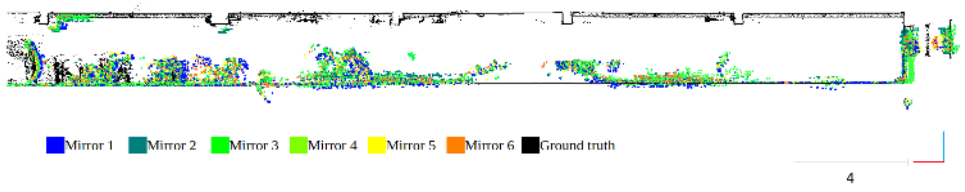
The mirror γ was represented by its plane parameters via $a_\gamma, b_\gamma, c_\gamma, d_\gamma$; the current rotation of the rotating table was represented with homogeneous transformation, $\begin{bmatrix} \boldsymbol{R}_\beta^{tp} & \boldsymbol{t}_\beta^{tp} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \in SE(3)$. Finally, the residual for the point pair $(\mathbf{P}_a^g, \mathbf{P}_b^g)$ was given by equation (4.12). Every found pair contributes a new residual. The number of equations creates an optimization problem. The equation (4.12) is differentiated automatically with respect to all optimized parameters, which were: $\mathbf{R}_l^t, \mathbf{t}_l^t, a_\varphi, b_\varphi, c_\varphi, d_\varphi$. The optimization problem is solved using the Levenberg-Marquardt algorithm. The cost function is modified with robust kernel to limit impact of outliers. The whole procedure is repeated until the solution converges using Ceres solver [2].

The calibration procedure on each scan site showed a significant improvement. Pre-calibrated data (with CAD initial calibration) showed multiple images of the floor in the scan. This is clearly visible in figure 4.14a. This situation changes with the calibration procedure and images converge into one 4.14b. The calibration method was validated using the map of the known environment. The environment used in this scenario was the same underground parking lot as in chapter 4.5. The obtained map, provided by the high-precision TLS and manual registration, was treated as ground truth.

Two calibration scenarios were tested. The first one used only a single scanning station, and the second one used multiple stations. Ground truth data was used for quantitative evaluation. Figure 4.15 shows the results for the first, simpler scenario, and figure 4.16 shows the results for the second one. The method introduces some outliers,



(A) Initial CAD-derived calibration of catadioptric system, registered against ground truth.



(B) Optimized calibration registered against ground truth.

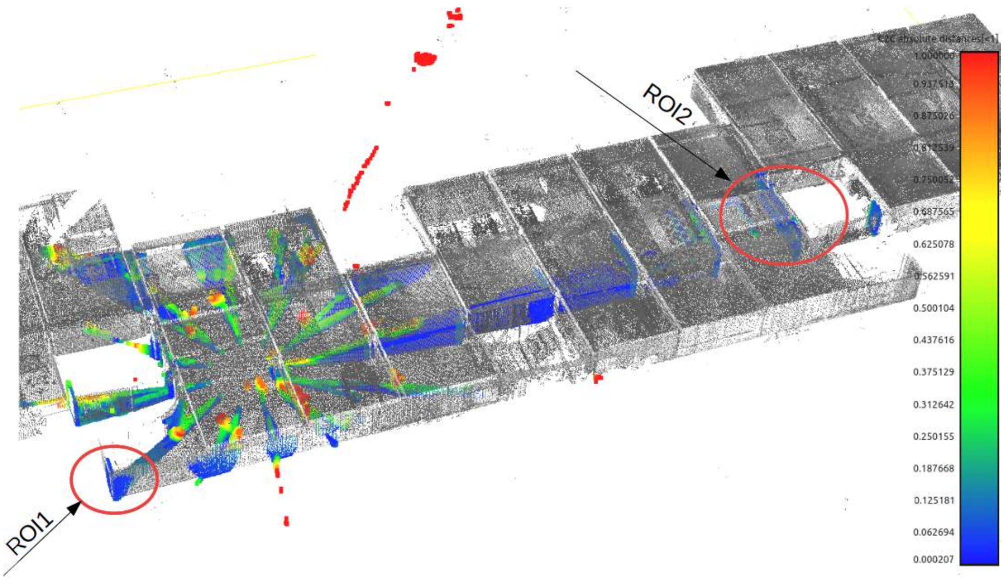
TABLE 4.2: The optimized parameters (planar reflectors' coefficients).

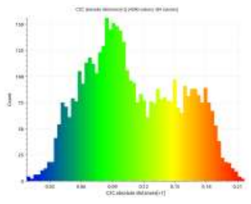
Parameter	Initial	Calibrated
$[a_1, b_1, c_1, d_1]$	$[0.793, -0.304, -0.527, -0.075]$	$[0.799, -0.311, -0.519, -0.044]$
$[a_2, b_2, c_2, d_2]$	$[-0.793, 0.609, -0.000, 0.075]$	$[-0.787, 0.614, -0.012, 0.049]$
$[a_3, b_3, c_3, d_3]$	$[0.793, -0.304, 0.527, -0.075]$	$[0.791, -0.301, 0.531, -0.049]$
$[a_4, b_4, c_4, d_4]$	$[0.793, 0.304, 0.527, -0.075]$	$[0.789, 0.311, 0.528, -0.049]$
$[a_5, b_5, c_5, d_5]$	$[-0.793, -0.609, -0.000, 0.075]$	$[-0.798, -0.605, 0.008, 0.045]$
$[a_6, b_6, c_6, d_6]$	$[0.793, 0.304, -0.527, -0.075]$	$[0.793, 0.291, -0.534, -0.047]$

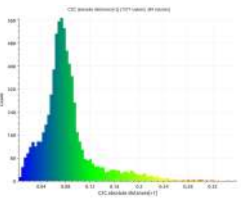
especially at the planar, horizontal structures near the sensor. It is clearly visible in 4.15. The vertical structures were represented accurately.

The distribution of error (figure 4.15 and 4.16) suffered from a larger numbers of outliers. They manifested as long-tail and positive skewness. The mode of error for a single station was 7.6 centimeters. For multiple scans, the mode of error distribution was 4.2 centimeters.

The amount of displacement which occurred during the presented calibration process is an interesting phenomenon. The exact values for all six reflectors are shown in table 4.2. The displacements of planes were minor. For example, the first planar reflectors rotated with calibration 0.65°.




(A) Histogram of errors for ROI1.


(B) Histogram of errors for ROI2.

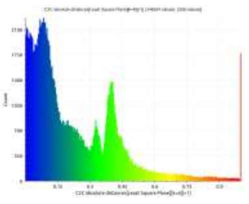
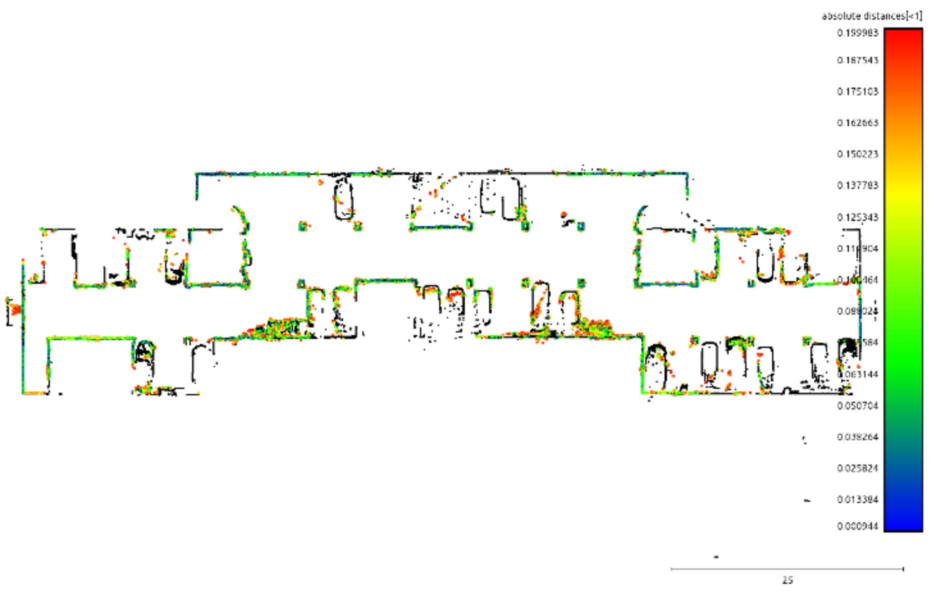
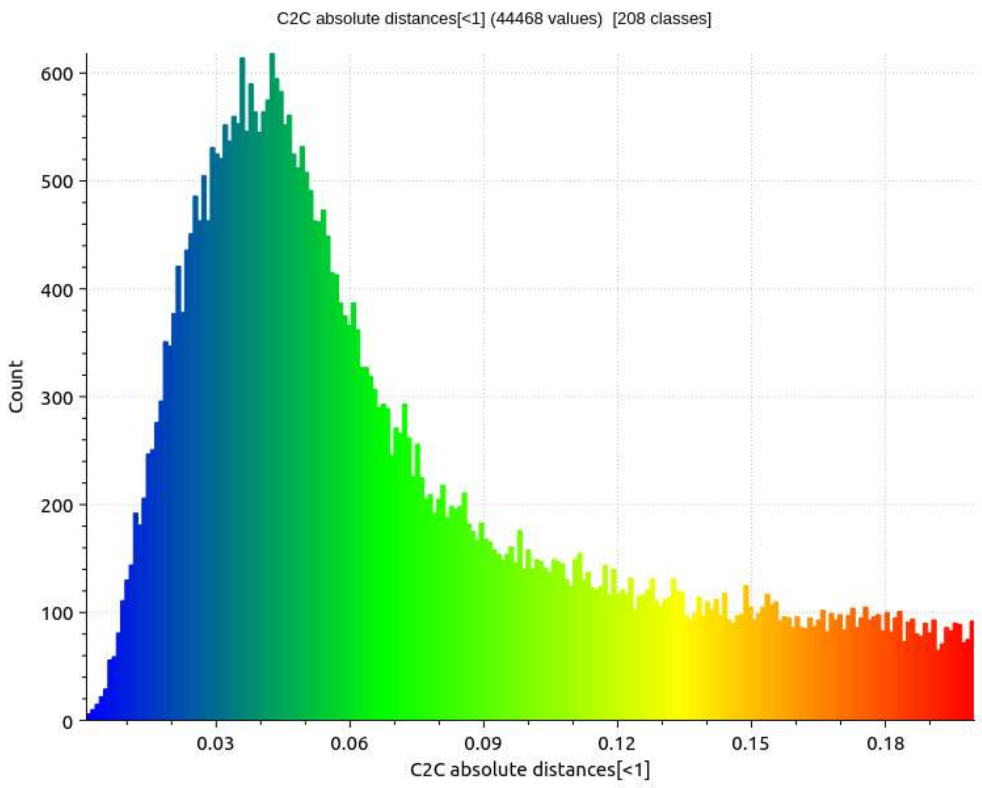

(C) Histogram of error for all point.

FIGURE 4.15: Result of calibration algorithm for single measurement station without ground truth data (simplest scenario).



(A) Comparison between investigated LiDAR and ground truth.



(B) Histogram of errors for all points.

FIGURE 4.16: Result of the calibration algorithm for multiple measurement stations with histogram of errors.

4.3 Calibration of a rotated reflector

The large number of outliers introduced by multiple planar reflectors discussed in previous section led to the research using only one planar reflector. This planar reflector can be rotated in a controlled manner to achieve a larger FOV. It is a similar approach to the 3D unit discussed in section 4.6 which rotates the Velodyne VLP-16. The whole assembly is simplified, as only a tiny and light planar reflector is rotated. This eliminates the need for the high-torque motor solution, bearings, and a slip ring. The shared components are hardware synchronization and motor driver. The hardware time synchronization is essential for achieving a precise measurement.

Time synchronisation with a Livox LiDAR

The Livox Mid-40 LiDAR allows for hardware synchronization by a PTP protocol or a PPS signal. The PPS signal is utilized in this design. This method of data synchronization is less trivial than in the Velodyne VLP-16 because the timestamp reported by the Livox Mid-40 LiDAR turnovers every second. Any system which synchronizes data must consider this solution for attaching the timestamp. The electronic system is identical to the one in chapter 4.4, but an extra algorithm for synchronization is introduced. The microcontroller reports timestamps effectively from 0 to infinity, which is the top plot in figure 4.17. The Livox Mid-40 LiDAR reports timestamps in range 0 - 1 second, which is the third plot from top in figure 4.17). It reports in a fractional part of a second and an arriving PPS pulse's rising edge resets its internal timer (the second plot from the top in figure 4.17). The program which collects data from the Livox LiDAR needs to reconstruct the whole part of the timestamp robustly. It is done with a helper register, as shown in the bottom plot in figure 4.17. The software running on a host machine, which performs data acquisition, collects two User Datagram Protocol (UDP) streams from the STM32 and the Livox LiDAR. The helper register is loaded with a value of the whole part of the timestamp strictly in between two PPS pulses. The host software observes the timestamp reported by the STM32 and loads the whole part of the reported timestamp in an exact moment, which is shown as the initial jump in the bottom plot in figure 4.17. Then, a falling edge is detected in the timestamp reported from the Livox LiDAR. The edge detector triggers when the last reported timestamp is close to 1 second, but the new one is close to zero. When such a case occurs, the helper register is incremented. The value from the helper register can be used as a whole part of timestamp, while the timestamp of a packet reported by the Livox LiDAR can be treated as fractional part of timestamp. In other words a sum of fractional timestamp reported in the package and a value stored in the helper register gives complete timestamp. In the operation, the whole part of the timestamp reported by the STM32 should always be equal to the value in the helper register in between two PPS pulses. If not, it means that the Livox LiDAR did not reset its timer on the PPS edge, e.g. due to a disconnection of synchronization wire.

The synchronization program is multi-threaded. The first task arranges incoming data from the STM32 into associated data which is ordered by the timestamp container, an implementation of a set from the standard C++ library. The data from the STM32 arrives with ascending timestamps, so inserting a new value to a set is computationally efficient. The thread also removes data older than 1 second. The second task arranges

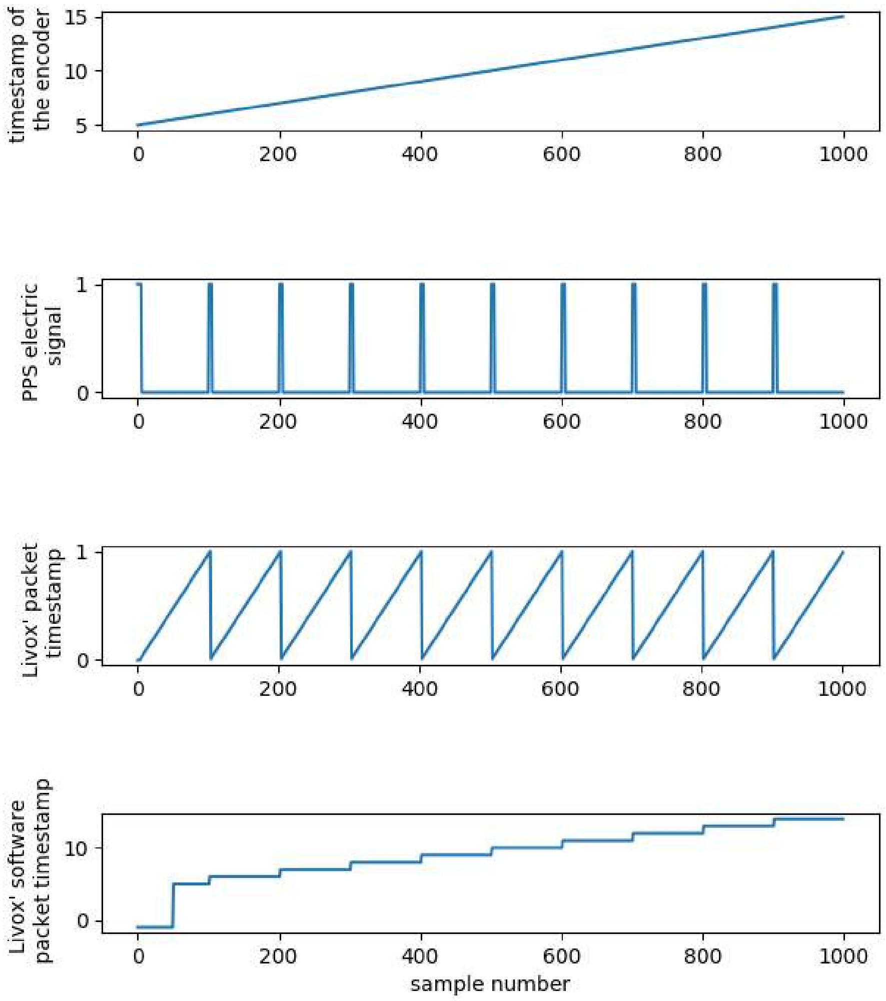


FIGURE 4.17: Synchronization registers and signals in synchronization solution for the Livox Lidar.

incoming data from the Livox Mid-40 into a sequenced container, an implementation of deque from the standard C++ library. It also reconstructs the whole part of the timestamp. The third thread pops data from a list of measurements from the Livox. It is done by keeping a minimum size of the list. The popped data from the front of the list has its timestamp, which is a key to query the ordered container of the packet reported by the STM32. With the query result, two nearest entries are found.

These two entries have the encoder's angle reported, so a bi-linear interpolation is executed to get the angle for the requested timestamp. Next, a geometrical transformation is done, and a packet is pushed to the output queue or a file. Mechanical design

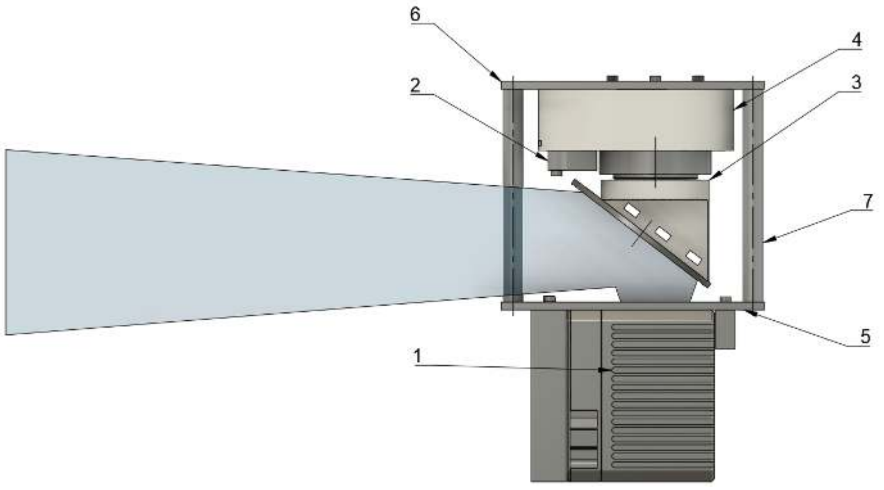
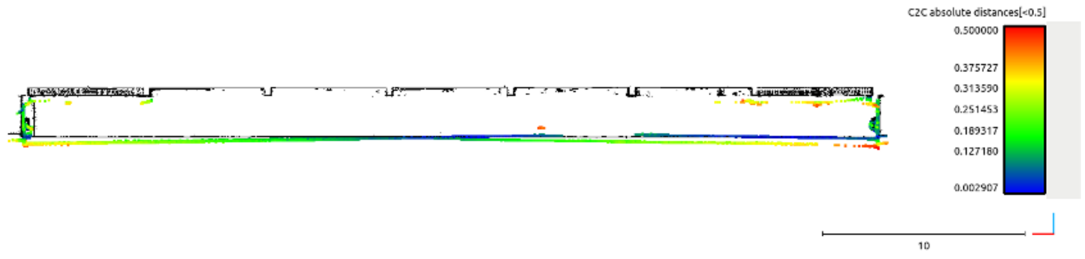


FIGURE 4.18: Rotating mirror assembly. 1-Livox Mid-40, 2- incremental encoder, 3- mirror support, 4- Motor housing, 6- top plate, 5-bottom plate, 7- pillars.

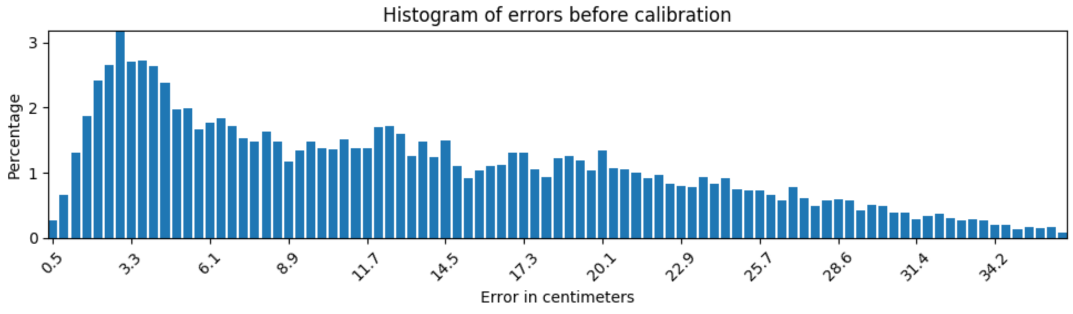
is presented in figure 4.18. The Livox Mid-40 LiDAR points to the direction of a tilted mirror which is attached to a motor. The motor is a BLDC sensorless motor with an incremental encoder attached. The mirror can spin at the speed of from 1 RPM to 200 RPM. The angle of the mirror tilt is constant, but it can be changed to obtain different shapes of the field of view similarly to figure 4.11.

Most of the parts were 3D printed or otherwise manufactured (e.g. CNC machined), but the whole assembly is far from perfect. As the software for data acquisition and transform requires a calibration of the system, the construction was taken to a known, previously mapped environment. The map obtained with a high-precision TLS was treated as a ground truth. The transformed results from the system were registered with the ground truth, and a histogram of error was built. If the calibration is derived from a CAD model, the results are considered poor. The error tail in left part of the error distribution is high (figure 4.19b).

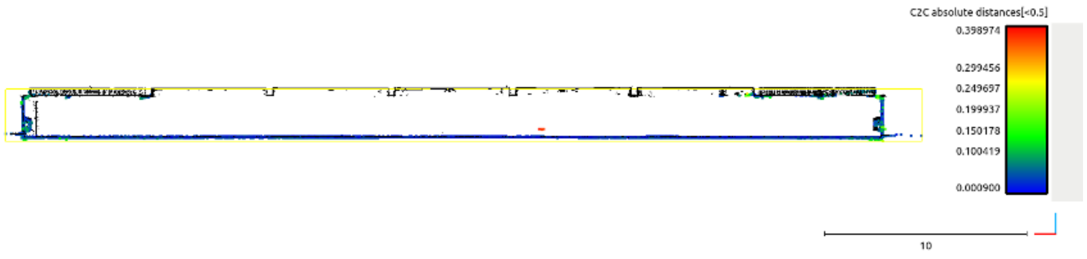
The calibration method used here is slightly different from the approach presented with multiple planar reflectors. As in the case of the multiple planar reflector, a rotation of the sensor is necessary for the optimization of the mounting transformation of the assembly on the turntable instead of the intrinsic parameters. The known environment and deflect calibrated parameters were used to minimize the error between the known map and the measurement.



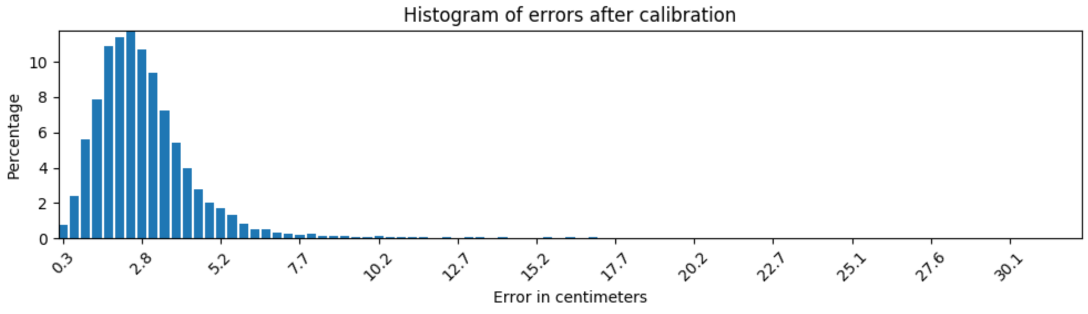
(A) Black-ground truth, other colors - input point cloud.



(B) Histogram of errors before calibration.



(C) Black-ground truth, other colors - input point cloud.



(D) Histogram of errors after calibration.

FIGURE 4.19: Quantitative evaluation of the calibration procedure.

It is a calibration method with a calibration field. Plane coefficients of the planar reflector and $SE(3)$ location of the rotation axis are geometrically optimized against the optical axis of the laser scanner. Two groups of degrees of freedom can accurately describe the imperfections in the assembly:

- reflector plane's coefficient $[a, b, c, d] \in \mathbb{R}^4$ describes the location of the reflector against the motor rotor, and it can compensate the angle of tilt, mirror thickness, and the encoder offset,
- $\begin{bmatrix} \boldsymbol{R}^c & \boldsymbol{t}^c \\ \boldsymbol{0}^{1\times3} & 1 \end{bmatrix} \in SE3$ of the rotation axis compensates the non-coaxiality and shift of the system, as well as it can compensate the encoder offset.

This description is somewhat redundant, and some of the degrees of freedom can act in the opposite direction. Qualitative tests of the redundant description against reduced plane (only mirror) show differences in favor of the redundant ones.

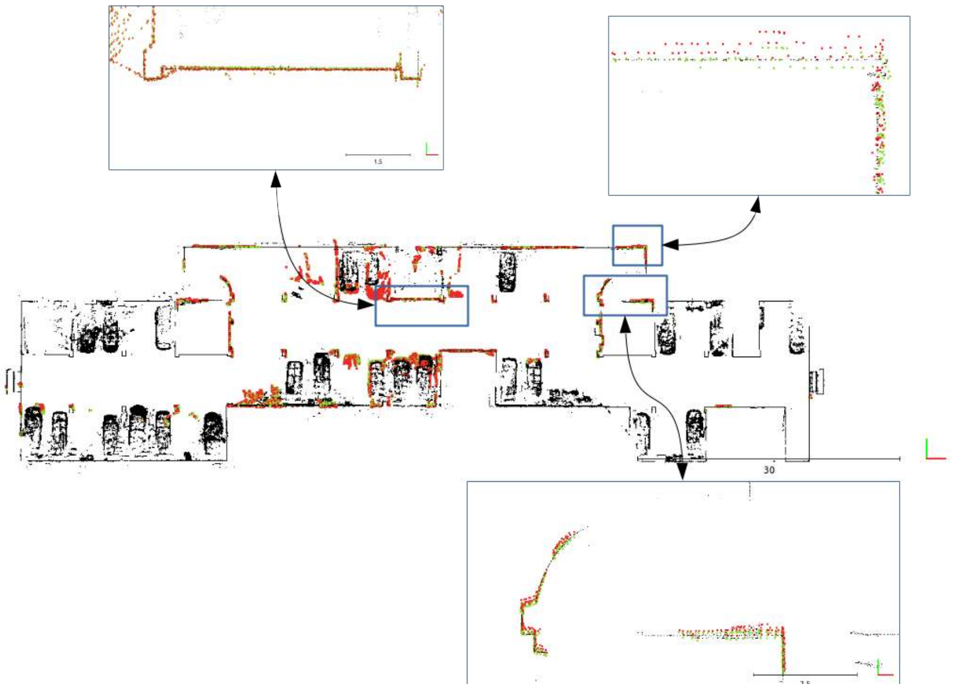


FIGURE 4.20: Comparison of the calibration results of the redundant model which optimizes $\left[\begin{smallmatrix} \boldsymbol{R}^c & \boldsymbol{t}^c \\ \boldsymbol{0}^{1\times3} & 1 \end{smallmatrix}\right]$, $[a, b, c, d]$ and extrinsic poses, and the reduced model which optimizes $[a, b, c, d]$ and extrinsic poses. The redundant model is drawn in green and the reduced one is drawn in red. The reduced model converges to a sub-optimal solution.

The calibration technique also required the optimization of the $\left[\begin{smallmatrix} \boldsymbol{R}^j & \boldsymbol{t}^j \\ \boldsymbol{0}^{1\times3} & 1 \end{smallmatrix}\right] \in SE(3)$ j-th pose of scanning stations. At least two scanning stations were used. A non-optimized parameter $\left[\begin{smallmatrix} \boldsymbol{R}^r & \boldsymbol{t}^r \\ \boldsymbol{0}^{1\times3} & 1 \end{smallmatrix}\right] \in SE3$ is a pure rotation reported by the encoder at the current moment of time. The plane $[a^r, b^r, c^r, d^r]$ is given :

$$\begin{bmatrix} a^r & b^r & c^r & d^r \end{bmatrix} = \begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^{c,r} & \boldsymbol{t}^{c,r} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}^{-1} \tag{4.13}$$

where :

$$\begin{bmatrix} \boldsymbol{R}^{c,r} & \boldsymbol{t}^{c,r} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{c} & \boldsymbol{t}^{c} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^{r} & \boldsymbol{t}^{r} \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \tag{4.14}$$

According to equation from (4.15) to further (4.18), the plane $[a^r, b^r, c^r, d^r]$ transforms a point in the Livox Mid-40 LiDAR's coordinate frame ($\boldsymbol{P}^l$) to a reflected point ($\boldsymbol{P}^r$).

$$\mathbf{r}^d = -2\left(\mathbf{b}^d \cdot \begin{bmatrix} a^r \\ b^r \\ c^r \end{bmatrix}\right) \begin{bmatrix} a^r \\ b^r \\ c^r \end{bmatrix} - \mathbf{b}^d \tag{4.15}$$

The vector $\mathbf{b}^d$ is direction of ray emitted by Livox LiDAR and $\mathbf{r}^d$ is direction of reflected ray. The intersection of plane $[a^r, b^r, c^r, d^r]$ with ray emitted by Livox LiDAR is given by equations (4.16) . The point $\boldsymbol{P}^{int}$ lays on a line created by vector $\mathbf{r}^b$: $\boldsymbol{P}^{int} = l^{int}\mathbf{r}^b$, where l^{int} is distance of point $\boldsymbol{P}^{int}$ from origin.

$$\begin{bmatrix} a^r & b^r & c^r & d^r \end{bmatrix} \begin{bmatrix} \boldsymbol{P}^{int} \\ 1 \end{bmatrix} = 0 \tag{4.16}$$

$$\begin{bmatrix} a^r & b^r & c^r & d^r \end{bmatrix} \begin{bmatrix} l^{int}\mathbf{r}^b \\ 1 \end{bmatrix} = 0 \tag{4.17}$$

Solving equation (4.17) for l^{int} and substitution to equation (4.16) gives intersection point $\boldsymbol{P}^{int}$. The point $\boldsymbol{P}^r$ is a point after reflection given with equation (4.18).

$$\mathbf{P}^r = -(\mathbf{P}^{int} + \mathbf{r}^d(l^p - l^{int})) \tag{4.18}$$

where: $l^p = \|\mathbf{P}^l\|$ is the length of the beam from its origin to $\mathbf{P}^l$, $l^{int} = \|\mathbf{P}^{int}\|$ is the distance to the intersection from the origin.

Finally, a global point for point Livox Mid-40 LiDAR's coordinate frame ($\boldsymbol{P}^l$) captured at pose j-th is given:

$$\begin{bmatrix} \boldsymbol{P}^g \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^j & \boldsymbol{t}^j \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}^r \\ 1 \end{bmatrix} = \psi^r\left(\begin{bmatrix} \mathbf{R}^j & \mathbf{t}^j \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, \begin{bmatrix} \mathbf{R}^c & \mathbf{t}^c \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, \begin{bmatrix} \mathbf{R}^r & \mathbf{t}^r \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, a, b, c, d, \boldsymbol{P}^l\right) \tag{4.19}$$

The function ψ^r entangles all previous operations :

- the transformation of reflector plane with equation (4.13),
- revealing of intersection point ray emitted by Livox with a plane equation (4.17) and (4.16),
- perform a reflected ray direction computation $\boldsymbol{r}^d$ with equation (4.15),
- recovery a reflected point location $\boldsymbol{P}^r$ with equation (4.18),

- a homogenous transformation of point $\boldsymbol{P}^r$ to global coordinate frame from j-th pose.

The point $\boldsymbol{P}^g$ is matched with point $\boldsymbol{P}^g_t$ in the ground truth map by the shortest distance. The cost of that pair equals:

$$\boldsymbol{P}^g - \boldsymbol{P}^g_t = \psi^r \left(\begin{bmatrix} \mathbf{R}^j & \mathbf{t}^j \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, \begin{bmatrix} \mathbf{R}^c & \mathbf{t}^c \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, \begin{bmatrix} \mathbf{R}^r & \mathbf{t}^r \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, a, b, c, d, \boldsymbol{P}^l \right) - \boldsymbol{P}^g_t \tag{4.20}$$

Introducing Lie algebra parametrization to equation (4.20), a optimized equation (4.21) can be build.

$$\underset{\boldsymbol{\sigma}_j, \boldsymbol{\sigma}_c, a, b, c, d}{\arg\min} \left[\psi^r \left((\exp(\boldsymbol{\sigma}_j^\wedge), (\exp(\boldsymbol{\sigma}_c^\wedge), \begin{bmatrix} \mathbf{R}^r & \mathbf{t}^r \\ \mathbf{0}^{1\times3} & 1 \end{bmatrix}, a, b, c, d, \boldsymbol{P}^l \right) - \boldsymbol{P}^g_t \right] \tag{4.21}$$

where $\boldsymbol{\sigma}_j \in \mathfrak{se}(3)$ is a Lie algebra map of j-th pose $\left[\begin{smallmatrix} \mathbf{R}^j & \mathbf{t}^j \\ \mathbf{0}^{1\times3} & 1 \end{smallmatrix}\right]$ and $\boldsymbol{\sigma}_c \in \mathfrak{se}(3)$ is Lie algebra map of rotation axis compensation $\left[\begin{smallmatrix} \mathbf{R}^c & \mathbf{t}^c \\ \mathbf{0}^{1\times3} & 1 \end{smallmatrix}\right]$. The system of equations (4.21) for all nearest neighborhood correspondences is collected. It is then optimized using LM algorithm. Because this algorithm is an iterative solution, the multiple iterations of the nearest neighborhood search, cost function building, and LM optimization are needed. The robust kernel are introduced to limit impact of outliers. The data is registered to the ground truth map with optimized intrinsic parameters when the whole solution converges similarly to ICP.

Results

Qualitative and quantitative results are shown in figure 4.19. Qualitative multiple image of floor visible in figure 4.19a scene are aligned together what is shown in figure 4.19c. The histogram of distances of uncalibrated is extremely skewed to right distribution (figure 4.19b). The error before calibration is expected to be large and distribution of observed error is flat. The histogram after calibration has much smaller righ-skew (4.19d). There is still no perfectly symmetrical Gaussian distribution, and small skew and fat tails in range of large error is visible. The source of those is mostly outliers, and occlusions. The expected error is about 2.8 centimeter up to 5.2 centimeter.

4.4 Calibration of a lightweight 3D unit

Mechatronic design

This unit rotates a LiDAR for reaching full FOV. For its advanced features, the motor controller needs sensor feedback, which is provided by the contactless incremental encoder. Typically a BLDC motors application needs to have a Hall's sensor which provides a square signal shifted in phase by 90 magnetic degrees to the current direction of the magnetic field. That signal is essential for the motor controller to shift energized phases of the motor.

In the presented solution Hall's sensors are absent. The motor controller allows simulating those from the current rotation position measured by the encoder. This

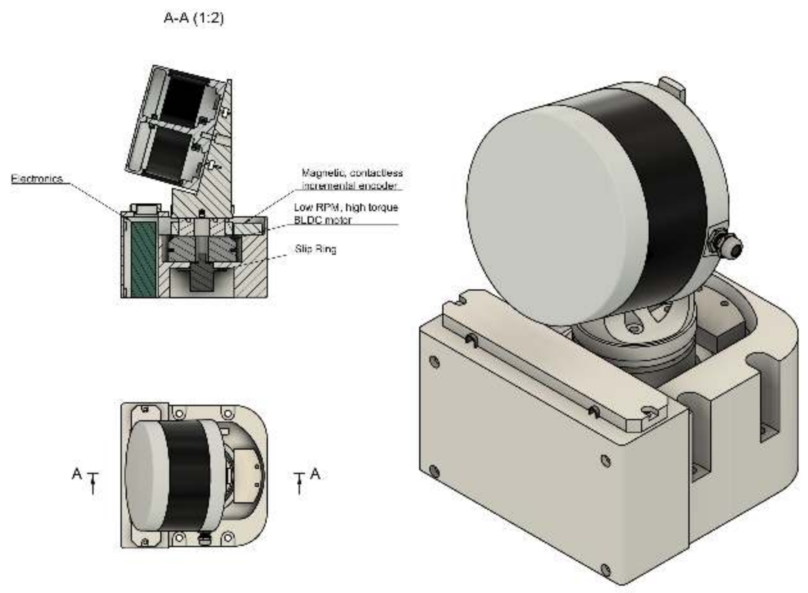

A-A (1:2)
Electronics
Magnetic, contactless incremental encoder
Low RPM, high torque BLDC motor
Slip Ring
A
A

FIGURE 4.21: Drawing of the lightweight 3D scanning system.

solution has a significant drawback, i.e. the Hall's sensor is a low-resolution, but absolute measure of an electric angle. However, the incremental encoder measures only relative rotation. To operate the motor controller, a particular procedure must be performed. The controller repetitively energizes motor coils with high current, similar to how a stepper motor is controlled. As a result, the motor is rotated rapidly. The controller measures the electromotive force on the idle phase and compares its derivative with the current location measured by the encoder. After a few cycles, the motor controller is entertained with the encoder-motor model. Finally, the motor starts rotating in a fully controlled manner and searches for a '0' tick in the incremental encoder, which is called homing. With the procedure complete, the motor is referenced and ready to operate.

The motor can be operated in three modes. The first one is a constant current mode, not used in this design. In it, the controller manipulates Pulse Width Modulation (PWM) duty cycle to stabilize current to a given set-point. In this mode a motor with no external torque limitation will spin to the maximum velocity limited only with voltage supplied. The second mode is a velocity controller. In this mode, the controller has two cascades, where the current setpoint is produced by a Proportional Integral Derivative (PID) controller that stabilize velocity. The motor will rotate with set velocity and react to external torque with increased current. Finally, the third and the most advanced mode of operation controls the position of the motor. In this mode, another control loop is added whose output is setpoint to the velocity controller. This mode allows following setpoint position accurately. With the limitation of electric current, this mode provides a robust solution which allows the head to be rotated externally or blocked without the risk of damaging the motor or the controller.

The main logic is designed in the STM32F1 chip. It is a simple, cost- and energy-effective Advanced RISC Machine (ARM) chip with multiple valuable peripherals. The firmware is developed in C language, with the following external libraries:

- STM Hardware abstraction layer which makes it convenient to access hardware peripherals of the STM32F1,
- LwIP (lightweight IP), an open-source network stack intended embedded system,
- author's CAN-Open stack introducing part of CiA 301 standard : Service Data Object (SDO) and Network Managment (NMT).

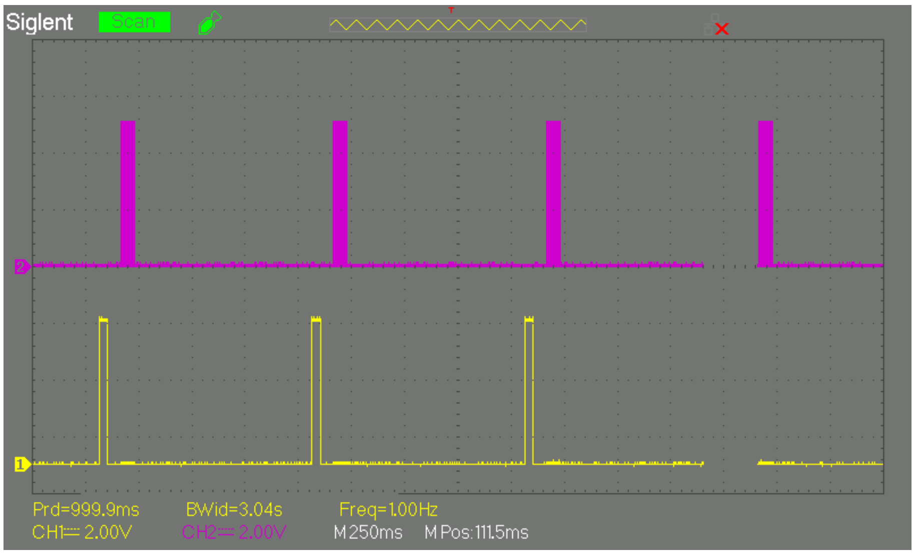


FIGURE 4.22: PPS (yellow) and TX (violet) signal show in oscillogram PPS signal is a short pulse with the frequency of 1 Hertz, TX is EIA RS-232C transmission with 9600 baud-rate containing 'GPRMC' NMEA message with simulated timestamp and position. Velodyne VLP 16 laser scanner is synchronizing its internal hardware timer to that signal.

The firmware is organized in a super loop design pattern, with heavily used interrupts. The super loop updates the non-time-critical tasks of CAN-Open communication, LwIP network stack, or Universal Asynchronous Receiver Transmitter (UART). Time-critical tasks, such as detection of zero crossings of incremental encoder or simulation of UTC second, are updated in hardware interrupts. The number of extra features is limited, but a further development would incorporate a real-time operating system which would provide better prioritization and advanced synchronization primitives. The chassis of the system is 3D printable or it can be CNC milled. The device drawing is shown in figure 4.21 and photo of whole assembly is shown in figure 4.25. A 3D printed chassis with modified polymer is preferable for low mass applications. The block diagram of system is shown in figure 4.23.

Calibration pattern

The calibration field consists of four square reflectors glued to a large plane. The reflectors are built using micro prismatic reflective sheets. Dimensions of the individual sheet are ten by ten centimeters. This material saturates laser scanner intensity measurements. The scanner itself provides correct distance measurement. The Velodyne laser scanner

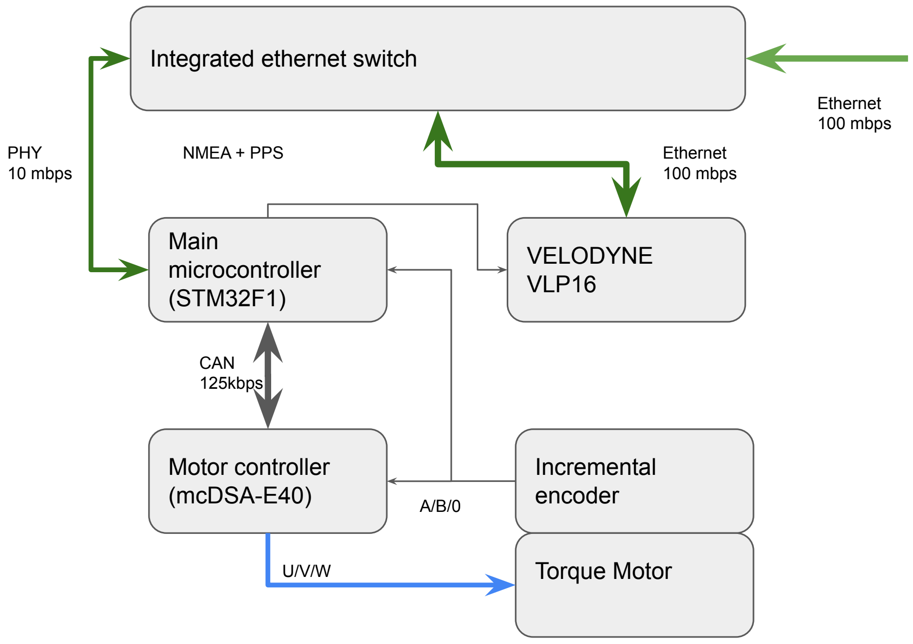


FIGURE 4.23: Block diagram of the system. Ethernet is marked in green, with different physical layers marked. The user has direct access to Velodyne VLP-16 TCP/UDP ports. Velodyne VLP-16 consumes synchronization messages from STM32F1 with pair of NMEA messages and PPS signal, that latches transmitted timestamp. Physically, NMEA message is sent on one TX line of EIA RS-232C with 9600 baud rate PPS signal is TTL square signal with frequency of 1 Hertz with a duty smaller than one. Oscillogram of such signal is shown in figure 4.22. The encoder output (quadrature pair with reference) is connected to STM32F1's timer-counter channels and feedback input of the motor controller. In this application a single encoder is used as measurement and control. CAN network of two nodes is marked in gray. Note that CAN network is used only inside the device, so the device's firmware implements only some segments of CiA 301 standard.

is an instrument intended for automotive applications, as it correctly handles surfaces with high reflectivity. In a similar design approach to Quick Response (QR) code or AR markers (fiducial), the contrast is maximized by a matte black background surface. The photo of calibration pattern is shown in figure 4.25.

Calibration pattern - LiDAR's intensity domain

While calibrating, the procedure system detects a collection of saturated points which lie on a single plane. At first, the step point cloud with an initial calibration is built. This results in a distorted image of the calibration field. The dual image of the calibration pattern is caused by the significant rotation of the Velodyne VLP 16 laser scanner around its central axis. A PCA/SVD pipeline whose purpose is plane fitting is executed for the found saturated points. Next, centroid $\boldsymbol{c}$ and the covariance matrix are found. The covariance matrix is decomposed using an SVD. The $\boldsymbol{V}$ matrix and the centroid are assembled into a homogeneous transformation matrix T_m.

$$\boldsymbol{T}_m = \begin{bmatrix} & & & c_x \\ & \boldsymbol{V} & & c_y \\ & & & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

The inverse of $\boldsymbol{T}_m$ transformation brings the calibration pattern to its local coordinate

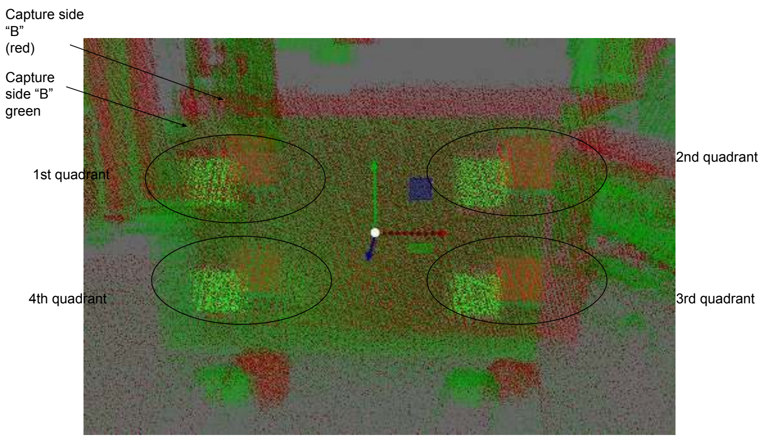


FIGURE 4.24: Detailed view of not calibrated field measured by the system. Bright colors correspond to the saturated points. Green points are the points that were measured by the left side of the laser scanner. Red points are points that were measured by the right side of the laser scanner (so-called side "A" and "B"). The drawn coordinate system that is visible in the middle is a coordinate system found by PCA/SVD of the saturated points, and the quadrant of XY sub-system are marked.

system, which is spanned on its principal directions (suitable plots in figure 4.24). Next, an orthographic projection is performed. Points that lie in given quadrants are the ones which obtain quadrant numbers. Detected points are organized into multiple sets. For a poor calibration of the system, there are two images of the calibration pattern visible (top right corner in the image 4.24). Points in the sets that were captured with side A do not align in the 3D space with points captured with side B. In total there are the following eight sets of points:

- $\boldsymbol{m}_{n1a}$ - n-th from N_{1a} saturated points of the first quadrant of calibration pattern captured by A side of the sensor,
- $\boldsymbol{m}_{n1b}$ - n-th from N_{1b} saturated points of the first quadrant of calibration pattern captured by B side of the sensor,
- $\boldsymbol{m}_{n2a}$ - n-th from N_{2a} saturated points of the second quadrant of calibration pattern captured by A side of the sensor,
- $\boldsymbol{m}_{n2b}$ - n-th from N_{2b} saturated points of the second quadrant of calibration pattern captured by B side of the sensor,
- $\boldsymbol{m}_{n3a}$ - n-th from N_{3a} saturated points of the third quadrant of calibration pattern captured by A side of the sensor,
- $\boldsymbol{m}_{n3b}$ - n-th from N_{3b} saturated points of the third quadrant of calibration pattern captured by B side of the sensor,
- $\boldsymbol{m}_{n4a}$ - n-th from N_{4a} saturated points of the fourth quadrant of calibration pattern captured by A side of the sensor,
- $\boldsymbol{m}_{n4b}$ - n-th from N_{4b} saturated points of the fourth quadrant of calibration pattern captured by B side of the sensor.

Cost function

The cost function measures the distances between the centroids of the corresponding sets observed with both sides of the sensor.

$$\boldsymbol{C}_1(\boldsymbol{T}_t) = \sum_{n=1}^{N_{1a}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n1a})\boldsymbol{T}_t\boldsymbol{m}_{n1a}}{N_{1a}} - \sum_{n=1}^{N_{1b}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n1b})\boldsymbol{T}_t\boldsymbol{m}_{n1b}}{N_{1b}} \tag{4.23}$$

$$\boldsymbol{C}_2(\boldsymbol{T}_t) = \sum_{n=1}^{N_{2a}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n2a})\boldsymbol{T}_t\boldsymbol{m}_{n2a}}{N_{2a}} - \sum_{n=1}^{N_{2b}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n2b})\boldsymbol{T}_t\boldsymbol{m}_{n2b}}{N_{2b}} \tag{4.24}$$

$$\boldsymbol{C}_3(\boldsymbol{T}_t) = \sum_{n=1}^{N_{3a}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n3a})\boldsymbol{T}_t\boldsymbol{m}_{n3a}}{N_{3a}} - \sum_{n=1}^{N_{3b}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n3b})\boldsymbol{T}_t\boldsymbol{m}_{n3b}}{N_{3b}} \tag{4.25}$$

$$\boldsymbol{C}_4(\boldsymbol{T}_t) = \sum_{n=1}^{N_{4a}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n4a})\boldsymbol{T}_t\boldsymbol{m}_{n4a}}{N_{4a}} - \sum_{n=1}^{N_{4b}} \frac{\boldsymbol{T}_r(\boldsymbol{m}_{n4b})\boldsymbol{T}_t\boldsymbol{m}_{n4b}}{N_{4b}} \tag{4.26}$$

$$\boldsymbol{C}(\boldsymbol{T}_t) = \sum_{n=1}^{4} \boldsymbol{C}_n(\boldsymbol{T}_t) \tag{4.27}$$

where: $\boldsymbol{T}_t \in SE(3)$ - the optimized configuration of laser scanner, $\boldsymbol{T}_r(\boldsymbol{s})$ - the rotation of the rotating head in the moment of capturing point given with $\boldsymbol{s}$. Finally, the sum of cost contributions from every quadrant from (4.23) to (4.26) is given with equation (4.27). Every cost is dependent on calibration $\boldsymbol{T}_t$ with parametrization (4.28) $\boldsymbol{\sigma}_t^{\wedge} \in \mathfrak{se}(3)$.

$$\boldsymbol{T}_t = \exp(\boldsymbol{\sigma}_t^{\wedge}) \tag{4.28}$$

The optimization problem can be stated as in equation (4.29).

$$\arg\min_{\boldsymbol{\sigma}_t} \boldsymbol{C}\left(\exp(\boldsymbol{\sigma}_t^{\wedge})\right) \tag{4.29}$$

The optimal calibration $\boldsymbol{\sigma}_t$ is found by applying LM method to equation (4.29) with automatic differentiation. There is no need for robust kernel introduction, since the data association is guaranteed.

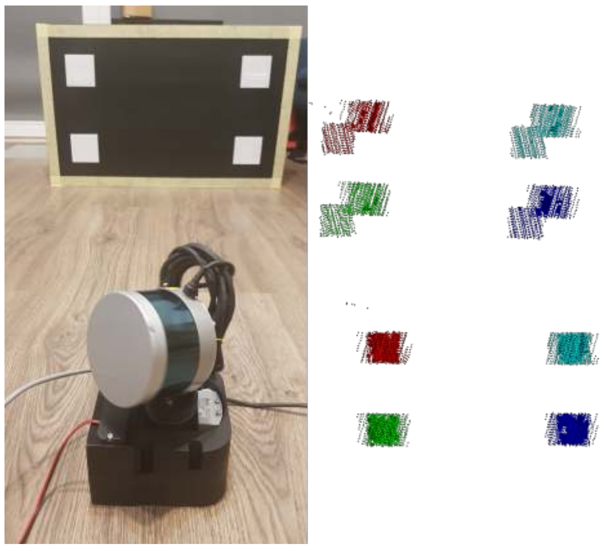
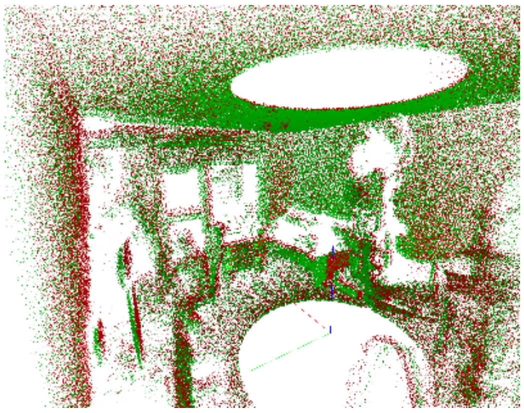
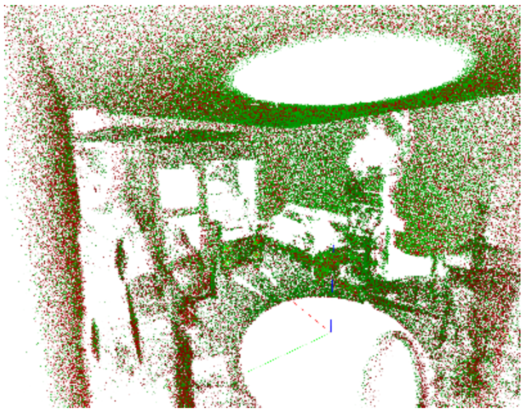


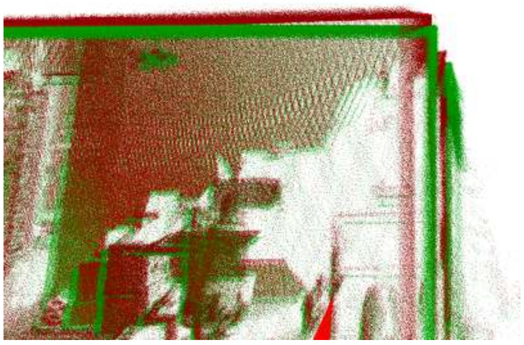
FIGURE 4.25: Left - photo of the calibration pattern, right top - projection of detected patterns using initial (CAD) calibration, right bottom - projection of detected patterns using optimized calibration.

Results

The algorithm effectively recovers geometrical correction. The image of calibration pattern becomes consistent after optimization what is shown in figure 4.25. The impact of the calibration for 3D pointcloud is shown in figure 4.26. The impact on calibrated parameters is shown in table 4.3. The distance from initial is 4.6 centimeters and rotation angle 15.4 degrees. The method is simple and robust but alters the extrinsic calibration. It can be further expanded with some knowledge about calibration scene (e.g. exact location of calibration marker against calibrated device).


(A) 3D pointcloud before calibration.


(B) 3D pointcloud after calibration..


(C) 3D pointcloud before calibration.

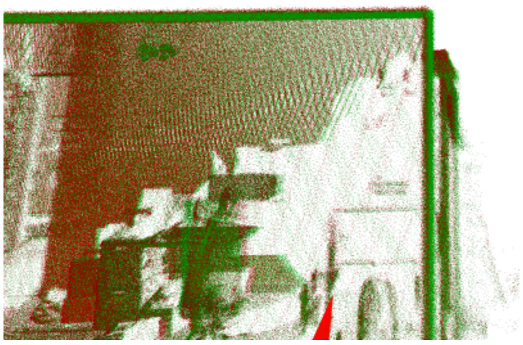

(D) 3D pointcloud after calibration.

FIGURE 4.26: 3D pointcloud before and after calibration. Red and green color represent sides of laser scanner.

TABLE 4.3: The optimized parameters before and after calibration.

Parameter	Initial	Calibrated
translation	$[0, 0, 0]$	$[-0.001, -0.011, -0.045]$
rotation vector	$[0, 0, 0]$	$[0.252, 0.047, -0.084]$

4.5 Calibration of a mobile backpack mapping system

This system consists of three Velodyne VLP-16 laser scanners in a rigid configuration. That rigid configuration includes:

- a top laser scanner whose main axis is horizontal,
- a side laser scanner whose main axis is vertical,
- a central laser scanner which is tilted by 30 degrees,
- a spherical camera in the top,
- an IMU.

The calibration of this system is challenging due to the minimal overlap of FOV of three individual lasers scanner. Two stages are performed to find extrinsic calibration of every device. The first stage recovers optimal trajectory of the movement system. The second stage finds extrinsic calibration of laser scanners. Both methods require data acquisition performed assuming slow and smooth motion, which is why the system is mounted on a manually operated turntable. The tight tolerances of the location, e.g. coaxiality of the rotation and main axes of the device, are not essential and not required. On the other hand, the rigidity of the whole assembly is essential for the calibration to succeed. The data acquisition procedure is simple: the user needs to start the data acquisition and manually rotate the crank until the backpack performs a full revolution. The system collects data from all of the sensors simultaneously.

The presented approach consists of two stages. In the first stage, a simple Graph SLAM problem is constructed to recover exact rotation movement during data acquisition. It is essential because of two facts. Firstly, the axis of rotation is not obtained with high precision and it can differ from the main axis of the calibrated system. Secondly, yaw measurement of IMU is noisy and somewhat unreliable. Due to the abovementioned facts, the initial trajectory is sub-optimal and needs to be further improved. In this Graph SLAM problem only data from two sensors, that is top LiDAR and IMU, is utilized. Resolving such a problem allows for recovering the trajectory of the movement of the system. An estimation of yaw angle is what is most challenging for IMU due to the gyro drift, but it is not a significant issue here. First of all, data acquisition is swift, and secondly, the IMU used here incorporates AHRS which allows for performing an absolute measurement of yaw angle. It is achieved by utilizing a magnetometer coupled with gyroscopes and an advanced real-time state estimator built into a device.

Optimization of trajectory

IMU is mounted at some distance from the rotation axis. Thus, a level arm is introduced. Hence, the initial trajectory has a circular shape, as seen in figure 4.7 in the top left. IMU introduces several hundred poses to be optimized in Graph SLAM. The problem is formulated using relative pose observation between consecutive poses. This mechanism allows for maintaining relative displacements between poses. Relative pose constraint is explained in section 3.2.2.

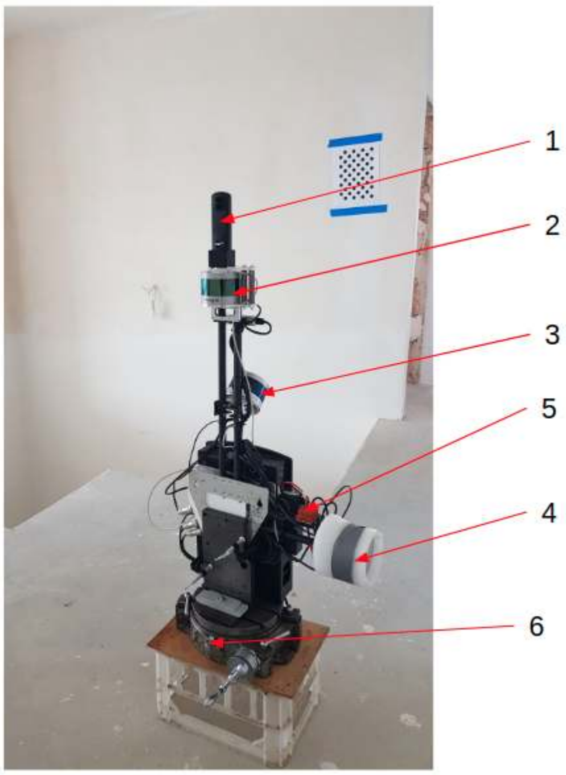


FIGURE 4.27: A mobile mapping system, mounted on the top turntable. 1-spherical camera, 2-top laser scanner, 3-center laser scanner,4-side laser scanner, 4-IMU, 6-side laser scanner (transport protection was not removed in the photo), 7-turntable with crank.

Every IMU reading contributes six parameters to optimization represented as $\mathfrak{se}(3)$ and six dimensions residual. The initial IMU measurement for time moment t is $\boldsymbol{p}_i(t) \in SE(3)$. Optimized pose for time moment t is $\boldsymbol{p}_o(t) \in SE(3)$. The orienta-

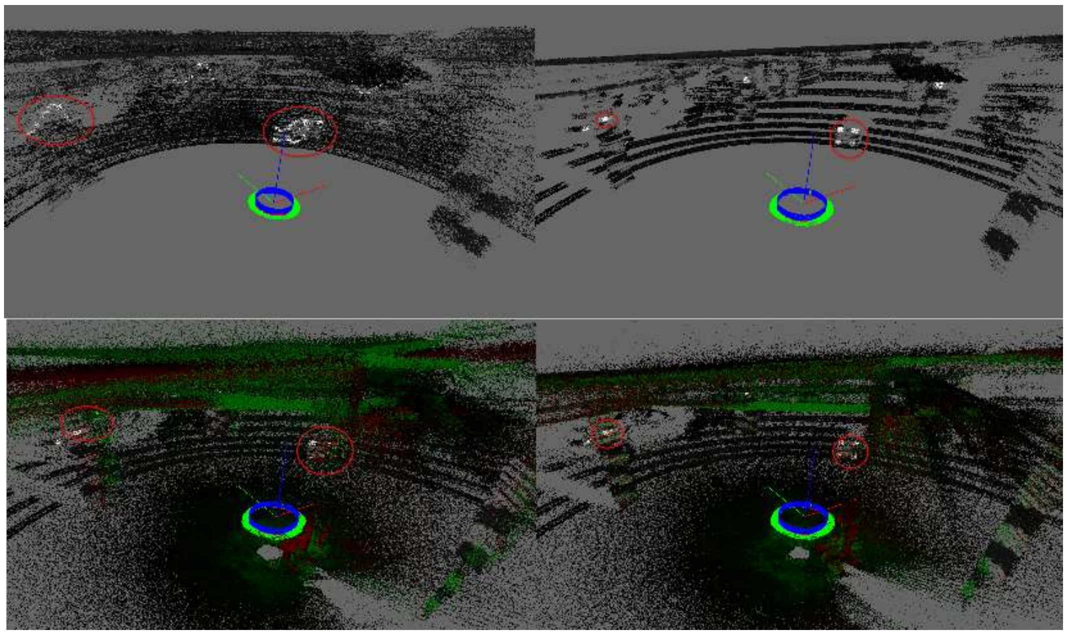


FIGURE 4.28: Steps of extrinsics calibration of a laser scanner. Top left - input data, visible only TOP laser scanner. Top right - the optimized trajectory of calibrated system. Bottom left - trajectory optimized, but non calibrated. There is a large visible discrepancy between the scans marked in green and red. Bottom right - the calibrated system. Control pads (retroreflector, marked with bright colors) are getting closer, which is a qualitative improvement.

tion measurements in time t and the subsequent measurement in time $t+1$ produce a residual in $\mathfrak{se}(3)$:

$$\boldsymbol{r}(t) = \log\left(\boldsymbol{p}_i^{-1}(t)\boldsymbol{p}_i(t+1)\right)^{\vee} - \log\left(\boldsymbol{p}_o^{-1}(t)\boldsymbol{p}_o(t+1)\right)^{\vee} \tag{4.30}$$

$$\boldsymbol{r}(t) \in \mathbb{R}^6 \tag{4.31}$$

The first pose is disabled from optimization. The second part of the optimization problem is related to the point cloud data from the top device. Due to its nature, the laser scanner observes the same scene from different view points. Here a standard point to point ICP metric from chapter 3.4.1 is used. In collected point-clouds sets of nearest neighborhoods are found. Every neighbor contributes one three-dimensional residual to the optimization problem:

$$\boldsymbol{r}(\boldsymbol{X},\boldsymbol{Y}) = \begin{bmatrix}\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times1}\end{bmatrix}\left[\boldsymbol{p}_o(t_x)\boldsymbol{c}(2)\boldsymbol{X} - \boldsymbol{p}_o(t_y)\boldsymbol{c}(2)\boldsymbol{Y}\right] \tag{4.32}$$

where: $\boldsymbol{X} \in \mathbb{R}^4, \boldsymbol{Y} \in \mathbb{R}^4$ are two points which are neighborhoods to each other, in homogeneous coordinate system. $(t_x, t_y) \in \mathbb{R}$ are the timestamps of point $\boldsymbol{X}$ and $\boldsymbol{Y}$

acquisition, $\boldsymbol{c}(2) \in SE(3)$ is the calibration of the top laser, treated as a constant at this stage. Note that optimized poses of trajectory $\boldsymbol{p}_o(t)$ for all time moments are represented in vector form of Lie algebra $\mathfrak{se}(3)$. Assuming that 1000 IMU measurements are taken and 10 000 neighborhoods are found, the system contains:

- 5994 optimized parameters (pose for timestamp t=0 is constant),
- $999 \cdot 6 + 10000 \cdot 3$ residual equations.

The problem is optimized using LM algorithm and automatic differentiation for the relative pose, and an analytical Lie algebra Jacobian for point to point distance minimization. After every iteration of the LM algorithm, the new set of nearest neighborhoods is found. As a result, the trajectory which came from IMU is refined. It mainly manifests itself with a change of the radius of rotation shown in the top row in the figure 4.28. Note that due to the high redundancy of the data (the same scene), only a handful of laser scans need to be processed, without any degradation of results. The intermediate poses will be "interpolated" by incorporating relative pose observation.

Optimization of an extrinsic laser scanner calibration

The laser scanners in the configuration shown in figure 4.27 have a minimal overlap with each other. This tiny overlap represent a challenge to automatic calibration. In the previous section, the optimal trajectory, which entails the least square distance of points in measurements collected by the top laser scanner, was found. This trajectory is used as a rigid transformation for a given point in time. Non-rigid transformations are poses of lasers between each other. This problem has twelve optimized parameters and six rigid ones (two $SE(3)$ poses of lasers scanners). A set of nearest neighborhoods is found between all collected scans.

This set contributes two types of residuals:

- pairwise,
- mutual.

The first type are observations taken in a different or exact time moment by a different laser scanner. They are residuals dependent on two poses, so its Jacobian is 12 by 3. The second type of observation is taken in the different time moments but with the same laser scanner. Its Jacobian is 6 by 3.

$$\boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y}) = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times1} \end{bmatrix} [\boldsymbol{p}(t_x)\boldsymbol{c}(\boldsymbol{X})\boldsymbol{X} - \boldsymbol{p}(t_y)c(\boldsymbol{Y})\boldsymbol{Y}] \tag{4.33}$$

where:
$\boldsymbol{X} \in \mathbb{R}^4, \boldsymbol{Y} \in \mathbb{R}^4$ - two points which are neighborhoods to each others in homogeneous coordinate system.
$(t_x, t_y) \in \mathbb{R}$ - timestamps of point X and Y acquisition.
$\boldsymbol{c}(\boldsymbol{X}) \in SE(3)$ - the pose of the laser which captured point X.
$\boldsymbol{c}(\boldsymbol{Y}) \in SE(3)$ - the pose of the laser which captured point Y.
If $\boldsymbol{c}(\boldsymbol{X}) = \boldsymbol{c}(\boldsymbol{Y})$, it was the same laser which captured X and Y. In this case, the Jacobian of residual equation (4.33) is smaller 12 by 3 version. This problem is optimized in a similar way as previously. Imperatively nearest neighborhood and LM is executed.

Quantitative and qualitative validation of extrinsic laser scanner calibration

The calibration was verified using a known, mapped environment. Additionally, retroreflector patches were installed in the environment. Retroreflectors are detected with significant intensity and are distinctive features in point clouds with intensity channels. For a well-calibrated system, these patches should create a consistent image with multiple lasers views. This effect is clearly visible in image 4.28.

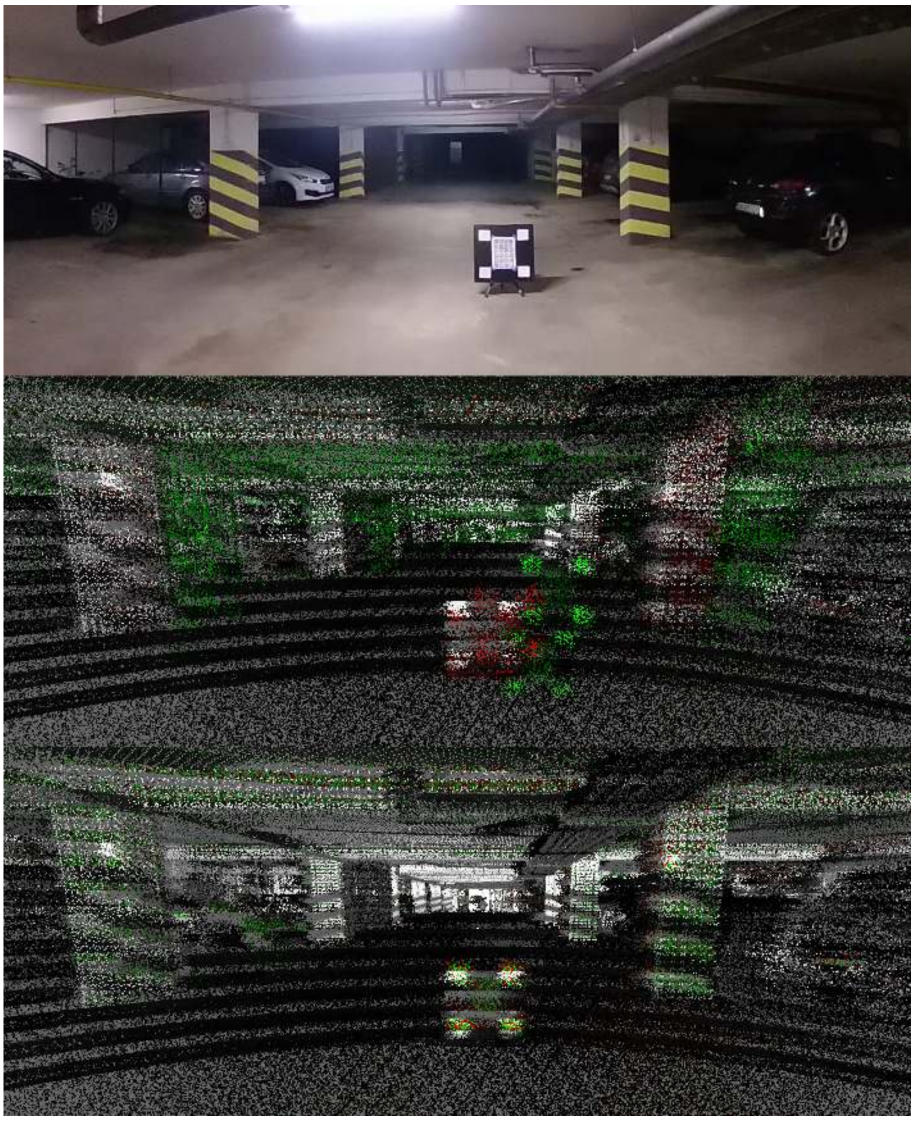


FIGURE 4.29: Calibration results. First row - camera image, middle row - CAD based calibration, bottom row - calibrated data. Green - center laser, red - side laser. Note that retroreflector images become more aligned along the process of calibration.

The surveyor company mapped the environment before the calibration was validated with a greater precision TLS (Z+F Imager 5010). That map is treated as ground truth. The results from the calibrated system are compared with the ground truth. First of all, the top laser was registered against ground truth. Simple point to point algorithm were used here. Nearest neighborhoods were collected between every laser scanner and ground truth. The distances of nearest neighborhoods were arranged in histograms, and both variance and mean for collected nearest neighborhoods were computed. The last row of figure 4.30 is a laser scanner used for registration of the system against ground truth. This laser scanner was rigid during the initial calibration. Both the first and the second laser scanner have significantly smaller mean and variances with the calibrated system. The uncalibrated lasers' histograms are inconsistent, while the calibrated lasers' histograms have similar mean and variance.

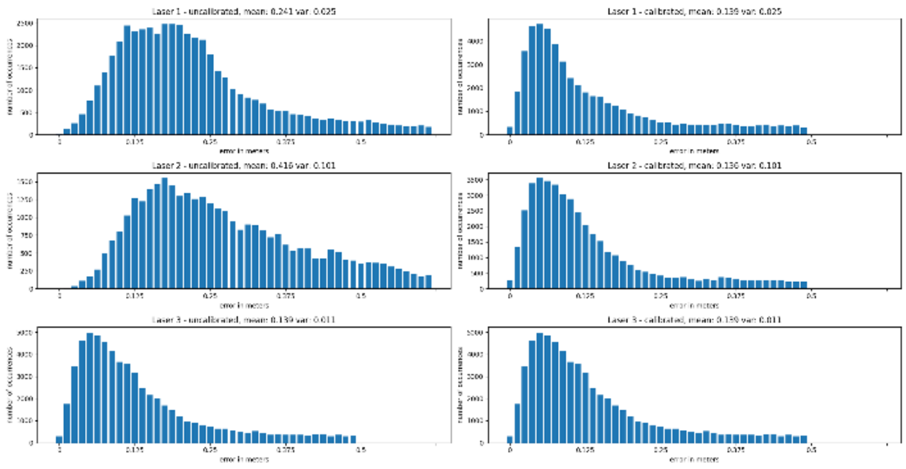


FIGURE 4.30: Calibration results. Left column - uncalibrated data, right column - calibrated data. The histograms build on top of evaluation of calibrated measurements are consistent in sense of variance and mean.

Note that the probability distribution shown in 4.30 is far from the Gaussian probability distribution. However, both are unimodal, non-symmetric, and positively skewed distributions. Also note that these histograms are long-tailed. This is caused by:

- the environment change between the moment of collecting ground truth data and validation data, e.g. different parking spots were used,
- noise caused by occlusions; some segments of the scene are reachable by only one laser scanner.

A common technique to eliminate these factors is hand-picking the desired control points from ground truth and using only them in the evaluation. According to the table 4.4, the system provides results with a median error of 7.5 centimeters. According to its manufacturer, the Velodyne VLP-16 LiDAR has an accuracy of 3 centimeters RMSE. Note that the total distance of the nearest neighbor for the laser number 3 (the top one) was minimized to find an optimal $SE(3)$ transform between the ground truth point cloud and the measurements. The median error is close to the ones found by other lasers, which indicates that the method provided an optimal solution to extrinsic calibration and confirms earlier qualitative observations.

TABLE 4.4: Comparison of median error to ground truth before and after calibration.

Laser number	Median dist. to groundtruth calibrated	Median dist. to groundtruth initial
1	7.57 cm	16.04 cm
2	7.75 cm	27.01 cm
3	7.59 cm	7.52 cm

Optimization of an extrinsic calibration of the spherical camera against a laser scanner

A spherical camera (component number one in image 4.27) is mounted on the top of the laser scanner (component number two in image 4.27). An approach which optimizes line features is proposed to cope with that problem. Line features can be detected precisely even in a sparse point cloud. The method used for this calibration is derived in chapter 3.5.1.

The calibration pattern which was used has three aluminum extrudes which contribute three lines to detection. The problem can be solved effectively when several observations are more significant than the number of the optimized parameters. It can be done by incorporating an observation from another pose. There is no need for knowledge of relative poses because the displacement between the camera and top Velodyne laser scan is rigid. These residuals can be minimized using automatic differentiation with the LM algorithm. In the sample capture, a structure was assembled using 40x40 mm aluminum extrusions arranged in a rectangular shape (left in image 4.31). These extrusions can be easily detected both in camera and LiDAR data. The user needs to manually mark the region of interest and trigger the algorithm which fits a line to the pointed regions. The result of this detection is shown in right in the image 4.31. To detect great circles projections in the spherical image, the user needs to mark more than two points in the picture which belong to the same line. The selected points are marked with stars in image 4.32. The projection obtained from initial calibration (from the CAD model) in shown in image 4.33. The initial calibration is sub-optimal and used as a starting value for further optimization using LM method. In this problem quality of the initial calibration is important, because the cost space has multiple minima. This is caused by the fact that the observation equation has two solutions, and only one is valid. That was explained in the chapter 3.5.1. It is clearly visible that projected curves miss the line features. Once the extrinsic calibration of spherical camera is optimized, the projected curves are aligned with line features.

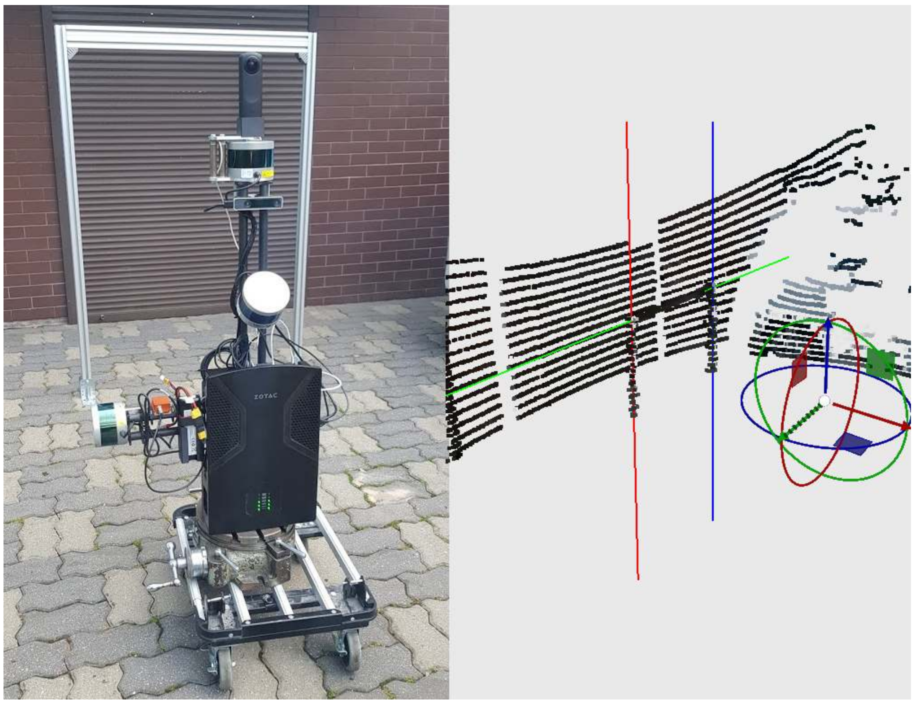


FIGURE 4.31: Calibrated system in left. There is a visible structure from extrusions that are easily detected by a laser scanner in the back. There is a point cloud captured by a laser scanner on the right with detected lines marked in colors. The gizmo manipulator shows the current state of calibration.

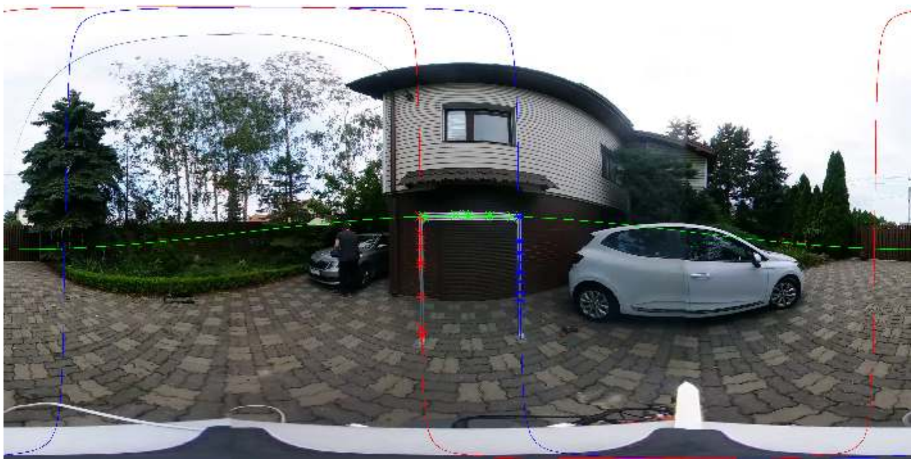


FIGURE 4.32: Input image. Please note stiching error changing with depth. The effect is similar to that shown in figure 1.6. Manually selected points are marked with stars. The next projection curve was fitted and plotted with lines.

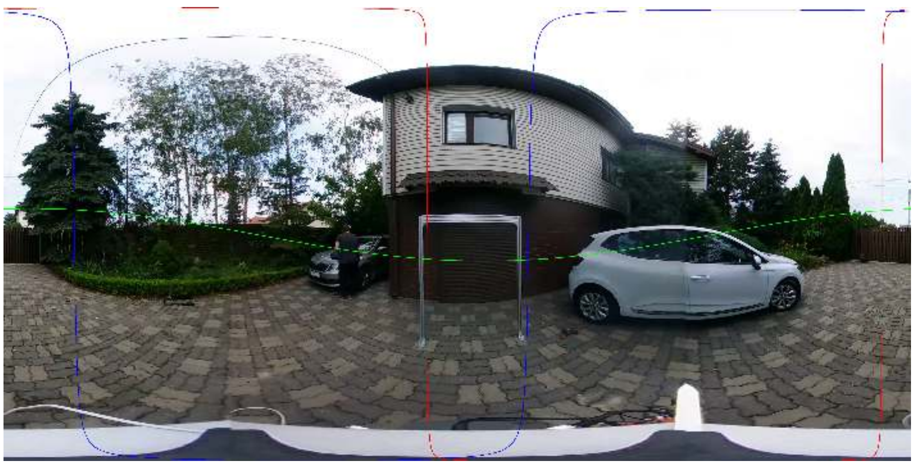


FIGURE 4.33: Pre-calibrated image with projected lines feature which were detected in the point-cloud.

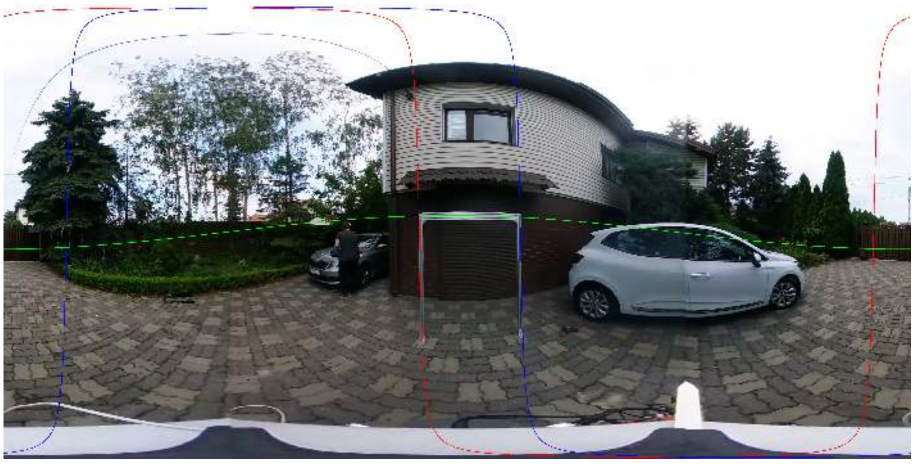


FIGURE 4.34: Post-calibrated image with projected lines feature that were detected in point-cloud.

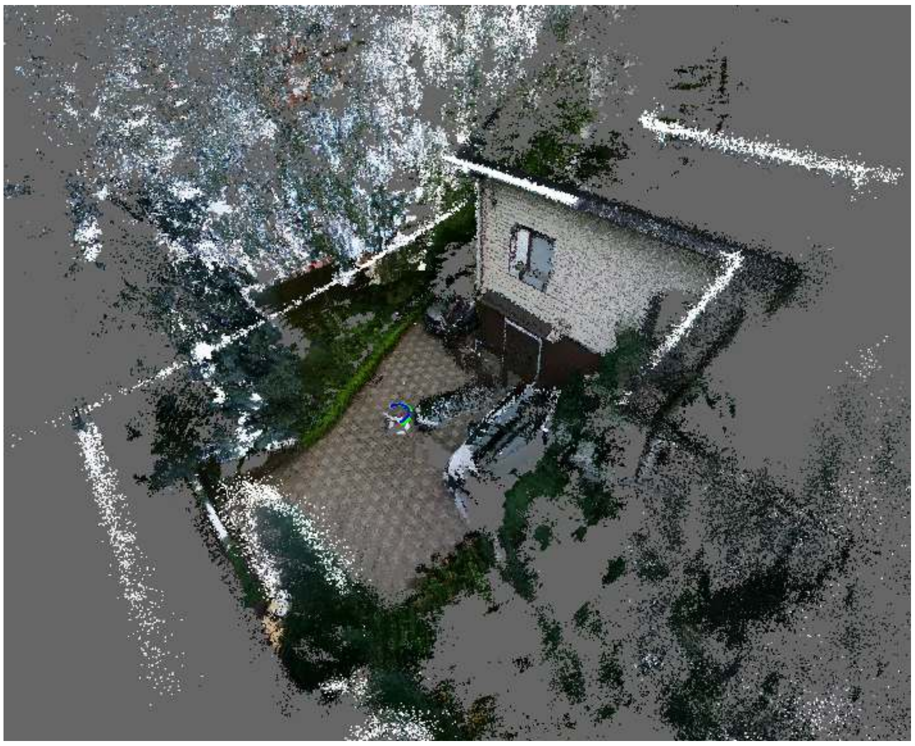


FIGURE 4.35: Calibrated extrinsic of spherical camera used for applying texture to point-cloud.

4.6 Calibration of a 3D unit

Mechatronical design

This system contains a Velodyne VLP16 laser scanner equipped with a Ladybug 5 camera. The Ladybug 5 camera is composed of six wide-lens global shutter sensors. This camera is widely used in land surveying and robotic applications [74]. The system in its design is a variation on the described lightweight 3D unit (section 4.4). It differs in regards to a rigid, low-play bearing, a high-resolution encoder, and a motor which enables a wide range of rotation speed. The design is waterproof (IP65) and tailored for a robotic mapping system. It consists of multiple system solutions:

- a construction able to withstand high shock,
- a construction able to withstand a wide range of mechanical vibration frequencies and amplitudes,
- a torque limiting mechanism,
- a dual encoder: a low-resolution encoder attached to the motor and a high resolution one attached to the main shaft.

The large rotating mass, which results in a high moment of inertia, has to be powered by a motor. Due to the cascading closed-loop control for motor and the geared reduction, the system can achieve a wide range of rotation velocity with precision. Unfortunately, the rapid rotation movement of the robotic base can cause the whole rotating head to generate a destructive torque for the gearbox and motor. To cope with this issue, a custom-designed torque limiting clutch was introduced. This mechanism allows for selecting a maximal allowable torque possible to transfer by the gearbox. Moreover, the clutch will release torque upon hitting an obstacle, thus preventing further damage.

The electronic design is almost identical to that presented in section 4.4 with two differences:

- the BLDC motor is equipped with hall sensors, allowing for the procedure for aligning motor's magnetic field with encoder to be omitted ,
- separated encoders are used for motor feedback loop and measurement.

The two designs share the same printed board circuit and firmware.

Calibration

Calibration of the system consists of two parameters which belong to $SE(3)$:

- a 6-DOF pose laser scanner,
- a 6-DOF pose of Ladybug camera.

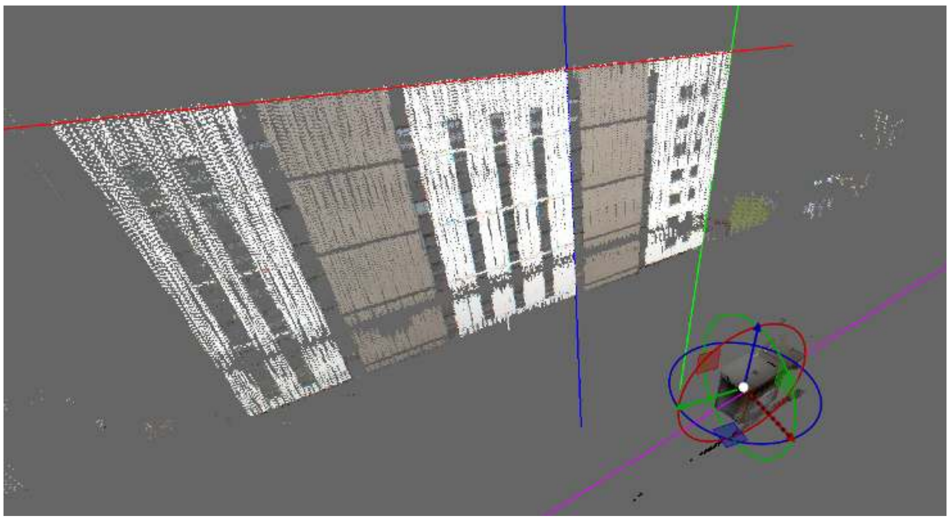
The Ladybug 5 camera is fully internally calibrated by the manufacturer. Extrinsic and intrinsic calibration parameters are available for every individual camera. The SDK provided by the manufacturer allows the user to create a spherical image with a different sphere projection radius. The SDK also allows for rectifying images and working on six rectified images directly.

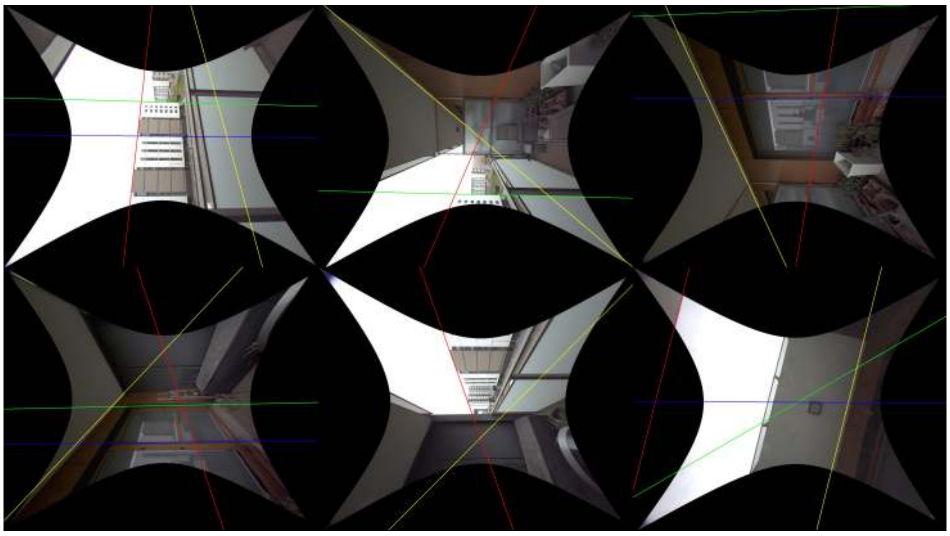
The calibration of the Velodyne VLP-16 laser scanner was conducted using a calibration field with retroreflectors in the same way as shown in section 4.4. The calibration of the camera was conducted using line features which apply the method shown in chapter 3.5.1. For calibration of the equirectangular image observation equation (3.130) was used. A modified version of the observation equation (3.134) that works on the rectified rectilinear image was used as well. The least square problem was designed from set of observation equations given in chapter 3.5.1. The sub-optimal calibration obtained from CAD model was used as initial condition and least square method yielded optimized parameters.

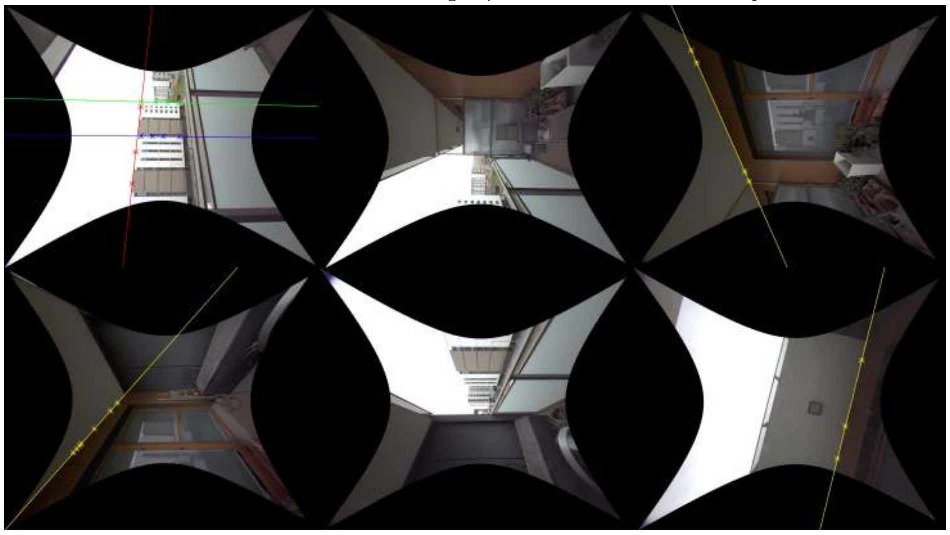
4.6.1 Comparison of the equirectangular and perspective model

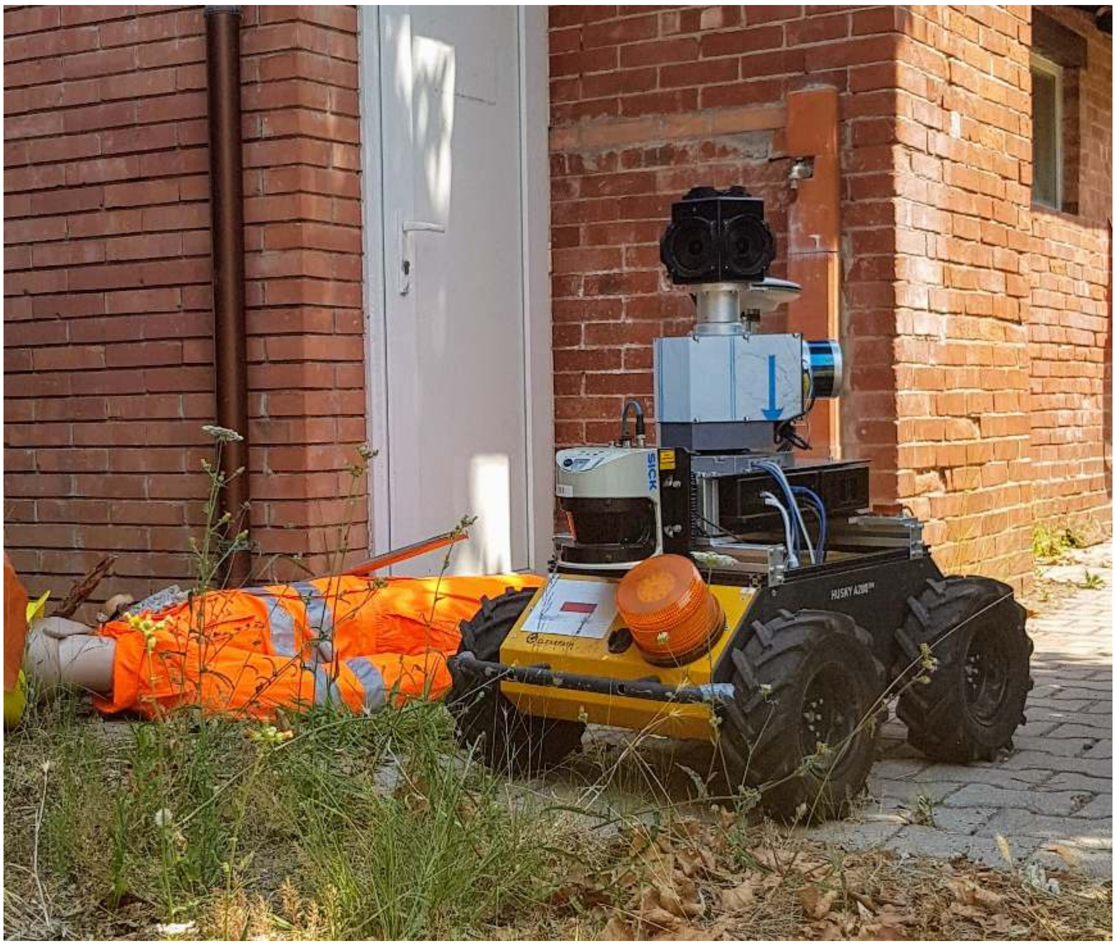
A major flaw of the equirectangular camera model was described in chapter 1.2.2. The model, which is parameterless and easy to use, can be calibrated effectively, but it suffers from simplification which causes a parallax error on depths smaller or greater than the sphere projection radius (e.g. shown in figure 1.8). This parallax error contributes to the discrepancies which can be noticed while comparing sample results as shown in figure 4.38. There is a misalignment of the intensity image and the color image in the far part of the scene (figure 4.38d) for a sphere projected on a radius of 2 meters. On the other hand, the close part of the scene has no misalignment (figure 4.38c). The opposite occurs for a sphere projected with a radius of 100 meters. The misalignment is visible in the close part of scene (figure 4.38e).

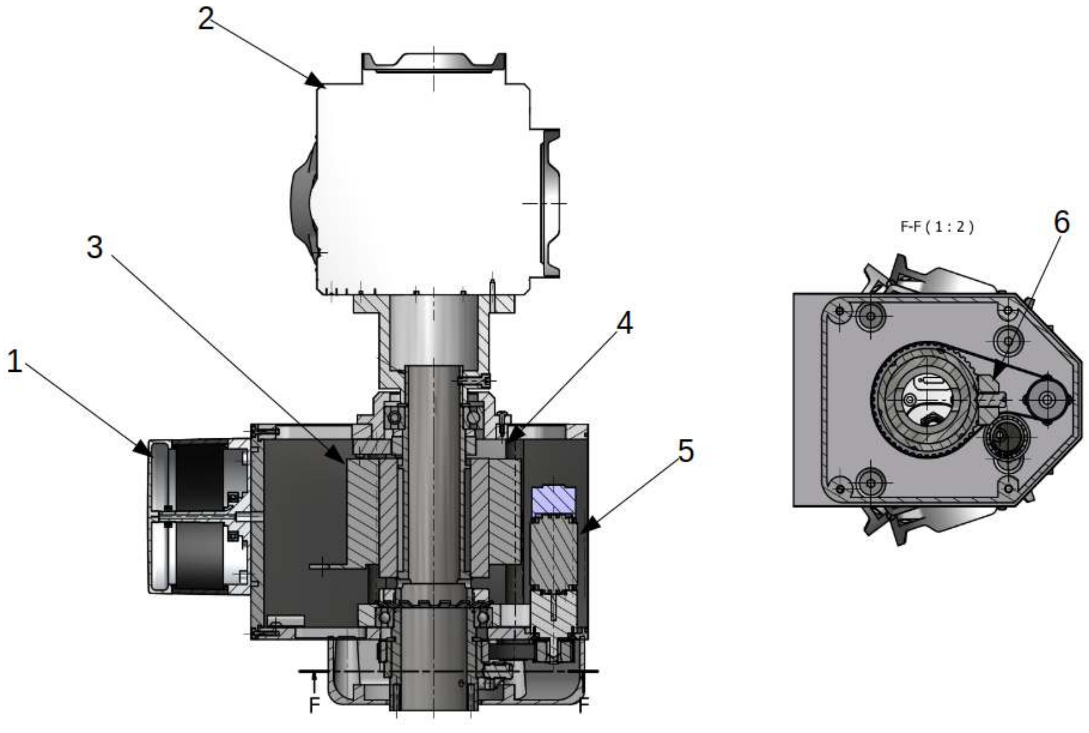
The perspective model is more consistent for both far and close objects (figures 4.38g and 4.38h). A similar observation can be inferred from figure 4.39. There is a clearly visible misalignment of the line feature with line in image when the depth of this feature is significantly different than the projected radius.


(A) Line features in 3D.

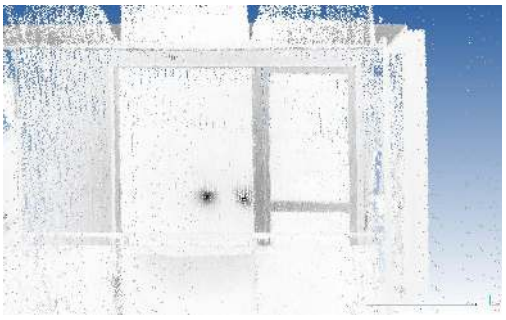

(B) Line feature from 3D projected on rectified images.

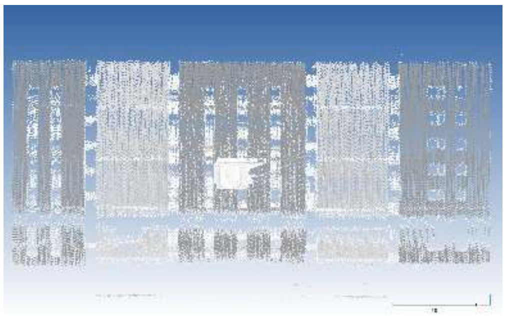

(C) Line feature in rectified images. Stars mark points used for the line detection.

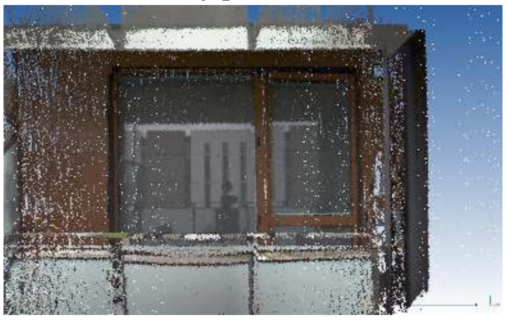

(A) 3D unit mounted on top of mobile robot during simulated search and rescue scenario.

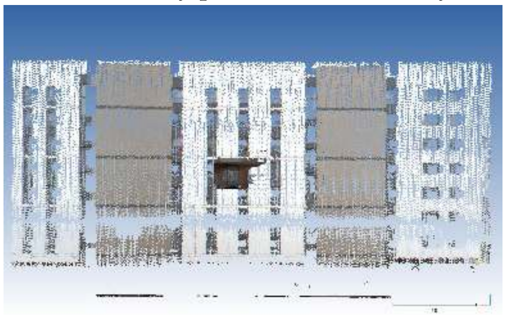


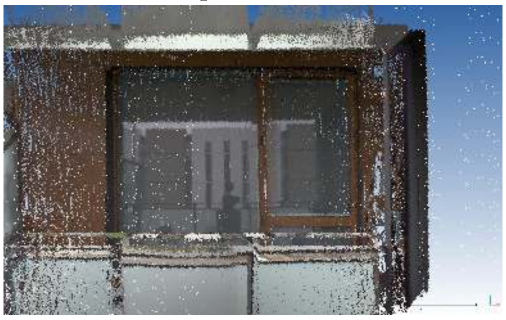
(B) Mechanical design. 1-VLP16, 2-Ladybug Camera, 3-Slip-ring, 4-High-resolution encoder, 5-Motor with low-resolution encoder, 6-Torque-limiting clutch.

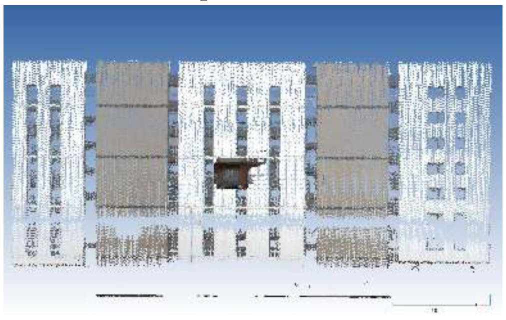

(A) Intensity point-cloud, close.

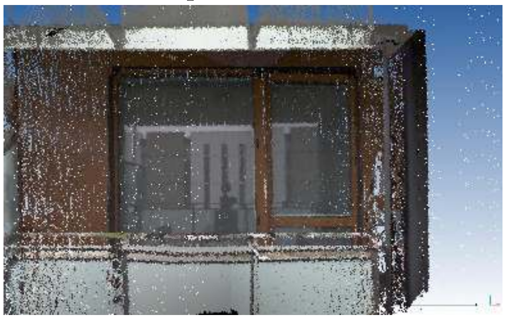

(B) Intensity point-cloud, far away.


(C) Image point-cloud, sphere stitched for sphere 2m.


(D) Color point-cloud, sphere stitched for sphere 2m.


(E) Color point-cloud, sphere stitched for sphere 100m.


(F) Color point-cloud, sphere stitched for sphere 100m.


(G) Color point-cloud, perspective stitched.

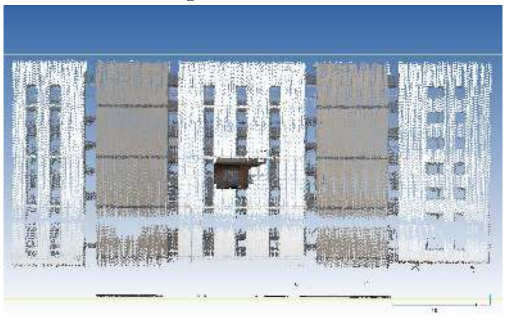
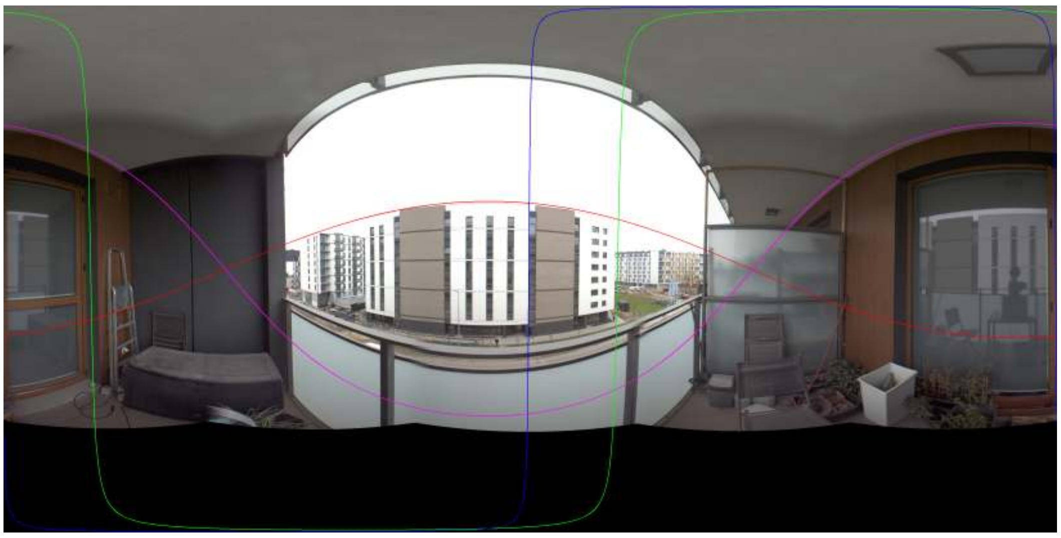
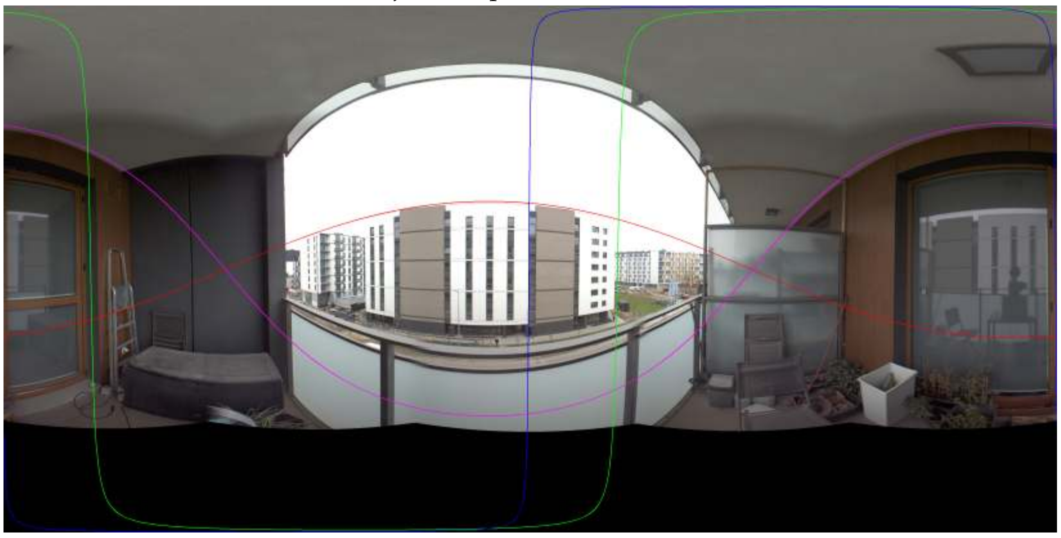

(H) Color point-cloud, perspective stitched.

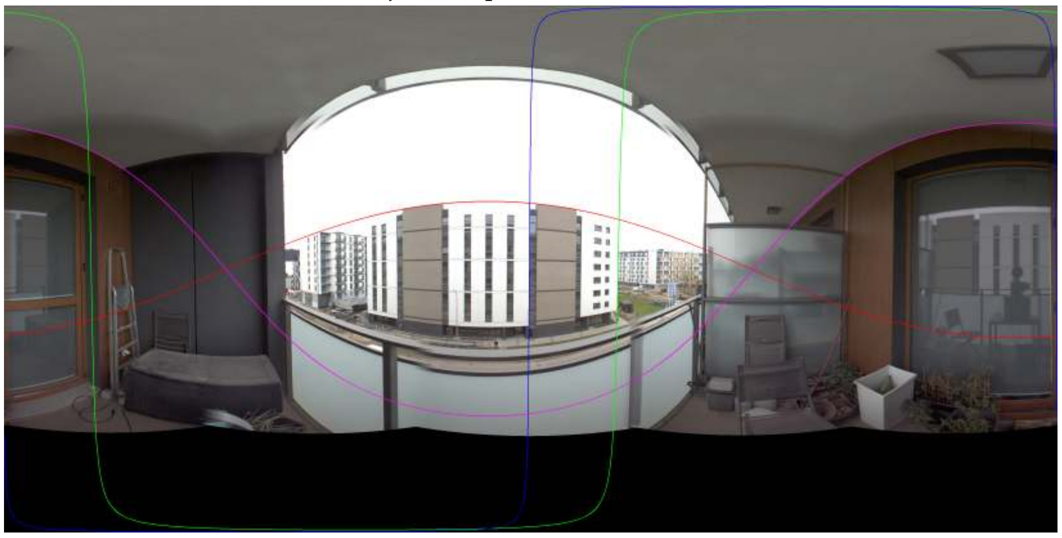
FIGURE 4.38: Color point-cloud, different stitching methods.



(A) Projection sphere radius 2 meters.



(B) Projection sphere radius 50 meters.



(C) Projection sphere radius 100 meters.

FIGURE 4.39: Projection of calibration line features on equirectangular image.

Chapter 5

Robotic applications

5.1 Mining shaft mapping

This system is designed for a specialized mapping application in a technical shaft in Wieliczka Salt Mine. The traditional approach to such a task is to perform a series of stationary scans using a TLS, but due to the limited access granted by the environment, it is not feasible for this challenge. Surveying work in such a shaft is expensive, time consuming, and it exposes the staff to the risk resulting from working on great heights and in areas of high pollution. The automation of mapping shafts reduces costs and the overall inspection time, and increase the safety of the staff. An inspection of a typical shaft with the depth of 650-1100 meters takes the surveyors from 24 up to 40 hours of what is essentially downtime, which in turn generates huge costs to the mine owners. The automation of shaft mapping can reduce the cost by at least 50% [1]. The mapped shaft described in this chapter is a technical one. There is no elevator with a cage fit to carry the surveyor and their equipment, so only a portion of the shaft can be accessed by this way. Therefore, these circumstances creates a unique opportunity to provide the mining industry with data not available before and without endangering the staff, which motivated the design process of a small mapping probe. The complete sensor solution comprises of:

- a rotated 3D unit with a Velodyne VLP-16,
- a tilted Velodyne VLP-32C,
- a Livox Tele-15 LiDAR,
- an AHRS IMU.

The sensors have been arranged closely so as to minimize the system's footprint, as shown in figure 5.1b.

This particular application is an excellent example of challenges that robotic mapping systems are facing. This particular application required a solution which was:

- rigid enough to survive collision with the mine installation,
- able to operate unassisted,
- equipped with an automatic calibration procedure,
- possible to be assembled and dissembled easily.

The assembled system has a build in hardware time synchronization. The rotating unit with the Velodyne VLP-16 provides the PPS signal for the rest of the system. The main computer runs a process to execute multiple threads collecting and synchronizing data and saving it as binary format. A wireless network access is provided with a web application for handheld device for the operator to enable them to check the system status.

The complete device is suspended on a cable attached to a winch. The winch is powered by an asynchronous induction motor coupled with a worm gearbox. The device's sensor setup is pointed downwards, as shown in figure 5.1a. The mapping process is conducted as follows:

- surveyors map the accessible parts of the shaft (e.g., its collar) using TLS with geo-referenced,
- next the system is suspended on a winch's cable,
- the data acquisition is initialized via a remote terminal,
- the operator of the winch carefully lowers the system with constant pace,
- once the system reaches the bottom of the shaft, the operator lifts the system,
- after the system is lifted the remote terminal is used to secure the data,
- once the operation concludes the system is dismantled.

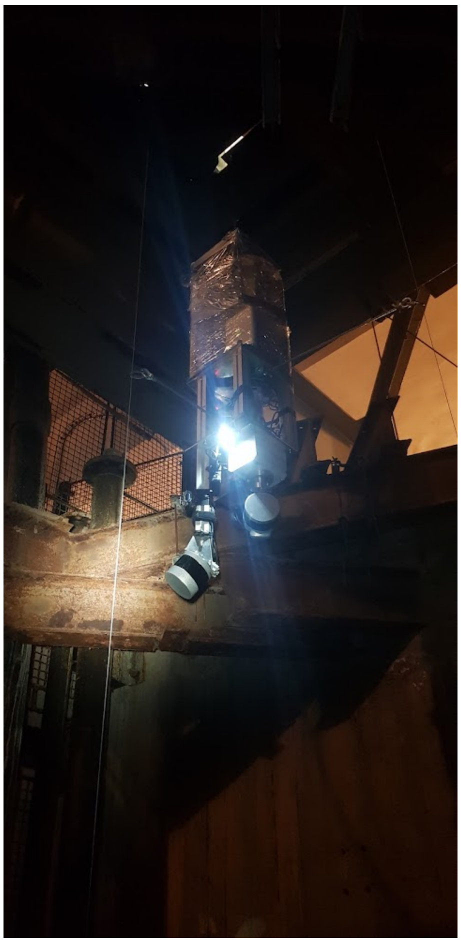
The initial trajectory of the mapping system is a linear movement recovered from the winch velocity and the rotation recovered by the AHRS IMU.

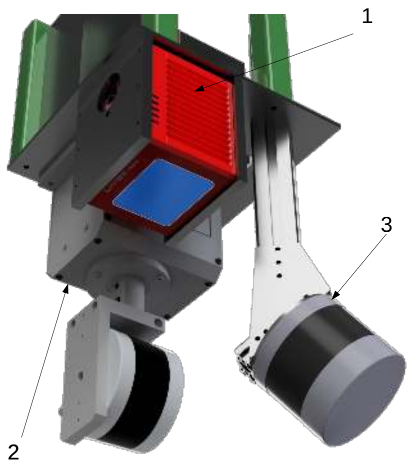
Calibration method

The shaft had been partially mapped using a precise TLS at multiple levels before these levels were reached by the mapping system. To obtain the calibration after the assembly of the mapping system the calibration procedure utilizing a high-precision map from TLS was used.

The FOV of individual sensors do not overlap. For it to obtain the correct calibration, the system needs to be moved and static environment assumption needs to be made. The calibration problem utilizes a trajectory constraint (see section 3.2.2) and a point to point metric (see section 3.4.1). Point to point observation equations are introduced for the nearest neighborhood points. Every observation equation tangles the optimized parameters. The nearest neighborhood search is performed between the high precision TLS scan and data from every sensor of the mapping system. The optimized parameters are:

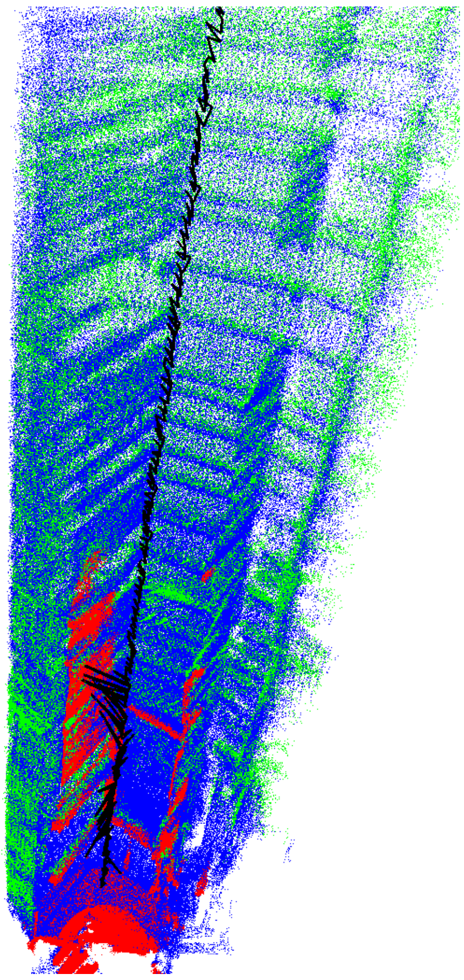
- trajectory nodes ($SE(3)$ every 0.1 second),
- calibration of every sensor (three optimized poses which belong to $SE(3)$),
- correction matrix against stationary scan (one optimized poses which belongs to $SE(3)$).


(A) Multi-sensor robotic mapping system suspended at the collar of mine shaft.



(B) Drawing of the multi-sensor robotic mapping probe. 1- Livox Tele-15 LiDAR, 2- scanning head with Velodyne VLP-16, 3- Velodyne VLP-32C.

FIGURE 5.1: Photo and drawing of the shaft mapping probe.


(A) Map of middle level, trajectory optimized without relative pose constraint.

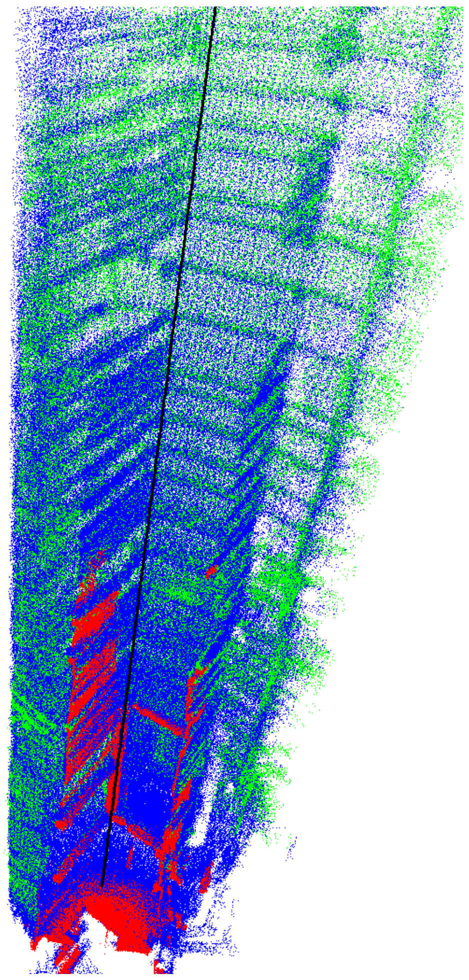
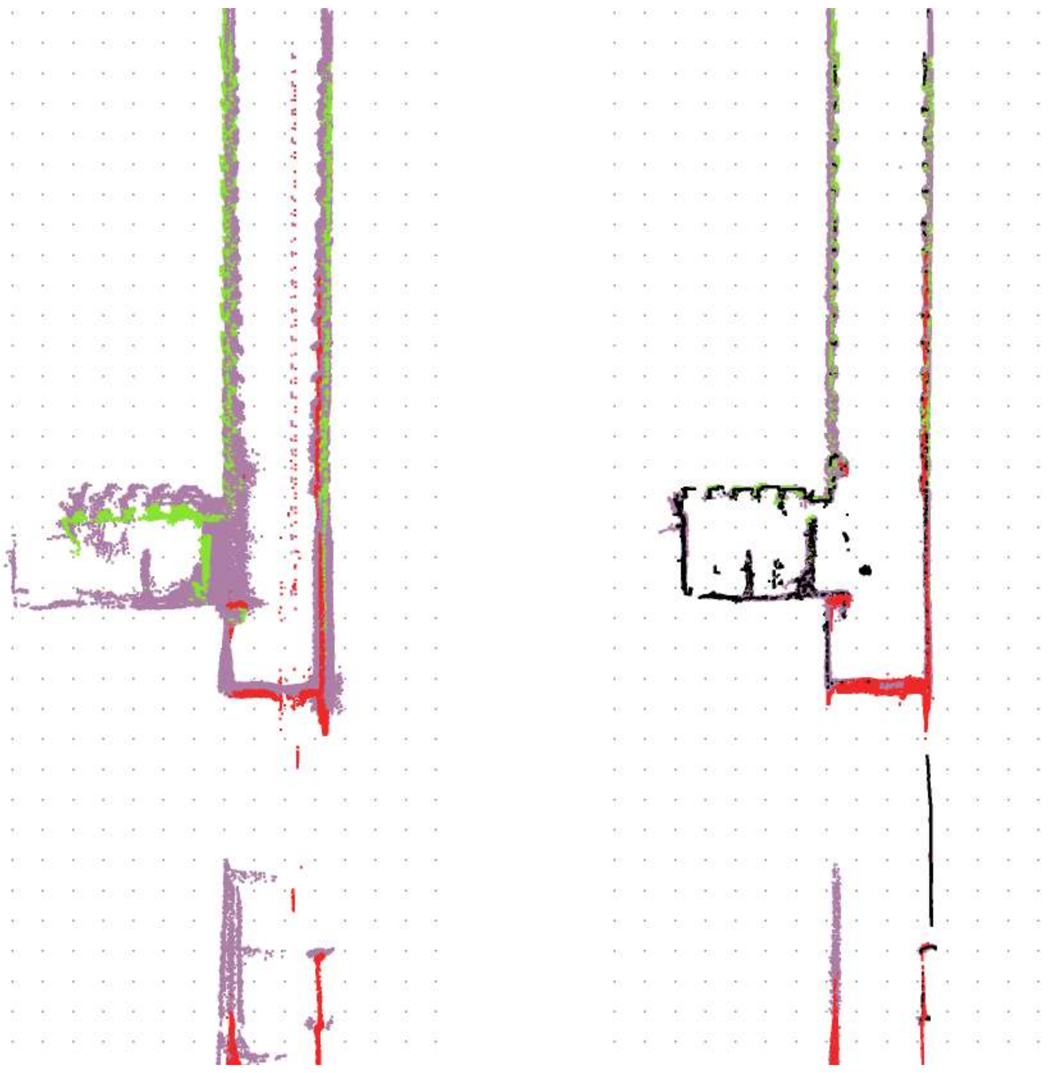
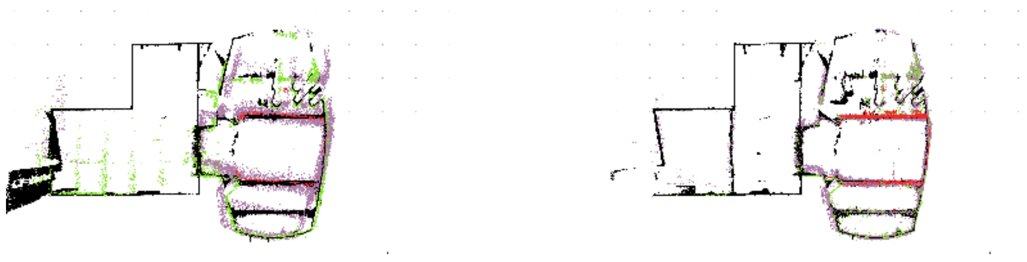

(B) Map of middle level, trajectory optimized with relative pose constraint.

FIGURE 5.2: Effect of trajectory constraint in automatic calibration. Both maps are registered, but differs in trajectory and recovered calibration. Left trajectory is jagged, right is smooth. To successfully perform an automatic calibration of such system, trajectory constrains are needed. Those constraints ensure that parameters regarding calibration will be optimized instead of individual trajectory nodes.



(A) Uncalibrated. (B) Calibrated.

FIGURE 5.3: Vertical intersection of shaft level, black - high-precision stationary scan, red - Livox Tele-15, green - Velodcyne VLP-32C, purple - rotated Velodyne VLP-16.



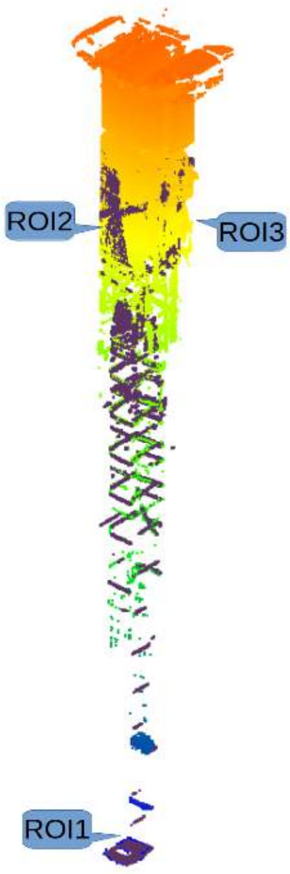
(A) Uncalibrated. (B) Calibrated.

FIGURE 5.4: Horizontal intersection of shaft level, black - high-precision stationary scan, red - Livox Tele-15, green - Velodyne VLP-32C, violet - rotated Velodyne VLP-16.

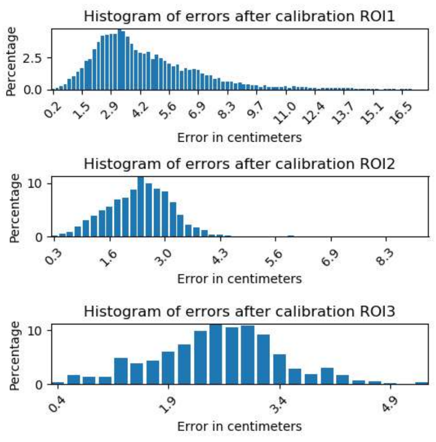
The need for trajectory constraints is shown in figure 5.2. The introduction of trajectory constraints is an attempt at limiting the SLAM problem to solutions which minimize optimized calibration. The pre-calibrated state of the system and the post-calibrated system against a calibration field is shown in figures 5.3 and 5.4.

Accuracy assessment

A procedure for quantitative assessment was implemented to verify the method of automatic calibration. As stated before, the environment had already been mapped in several places with a TLS. To assess the accuracy of the mobile mapping system, a calibrated scan performed by the device was registered against the result of stationary TLS. Short trajectory of about two seconds was used in this optimization problem for the accuracy assessment. The trajectory used for the calibration was 120 seconds. Three ROIs were selected and histograms of distances to the nearest points in the stationary scan were build. The ROI1 5.5b was observed from a distance of about 50 meter and it yielded an error not greater than 10 centimeters. This large error resulted from localization difficulties and the large leverage of other observation. Other ROIs yielded a smaller error. There is no visible static error introduced by the calibrated system. Error in ROI2 and ROI3 is comparable with the Velodyne VLP-16 precision [130].



(A) High precision stationary scan against calibrated scan (violet). ROIs are marked.



(B) Histogram of errors. The error on ROI1 is smaller than 10 cm. The error on ROI2 and ROI3 is smaller than 3.4 cm.

FIGURE 5.5: Accuracy assessment of calibrated robotic mobile mapping system against high-precision stationary scan.

5.2 Nuclear Power Plant Mapping

These mapping exercises were carried out as part of three editions of the European Robotic Hackathon, which took place in Zwentendorf Nuclear Power Plant in Austria [123] in 2017, 2019, and 2021. The most significant challenges in this benchmark were:

- an unpredictable environment (positive and negative obstacles, stairs, low light),
- a limited time for exploration tasks (30 minutes),
- a limited time for data processing (15 minutes),
- the robotic mapping system operating in the vicinity of radiation sources,
- no line of sight communication with the robot.

The challenge called for a very robust robotic mapping system. What is more, the limited communication bandwidth made teleoperation impossible, so partial autonomy was needed.

During the event, the operator was presented with a traversability map rendered using the robot's odometry and 3D mapping system, shown in figure 5.7. If necessary, the operator could obtain a spherical image providing them with important insight. The operator's task was to navigate the robot to the points of interest and to perform a scan there. When the system reported the completion of data acquisition, the operator could navigate to the next point of interest. When the robot's mission was finished, the operator began synchronizing data with the base station. After a few minutes, a map registered using robot's odometry and the onboard SLAM was transferred. Next, the operator was able to start a process that performed additional refinement of the map, resulting in map (5.8).

The second mode of operation to be performed by the system was mobile mapping. In this scenario, the robot moved continuously. This mode did not provide the operator with a color layer, and it had limited precision, but it could cover a larger area. Contrarily to how it was done before, the operator did not stop the robot during scanning in this mode.

Due to a larger amount of data, a different approach was used for data registration, which called for a factor graph SLAM being built. The robot aggregated data for a short period of about 2 seconds, which is the time required to perform a full revolution of the scanning head. In the aggregation step, the incoming synchronized points from the rotated laser scanner were transformed by the EKF estimation of the robot's pose. Scans were stored in their local coordinate system, which was the pose of the robot when the aggregation had started. The system stored aggregated scans and poses for optimization.

Two SLAM algorithms were used. The first is a factor graph SLAM. The hidden variables are $SE(3)$ poses of the robot. Multiple factors were introduced to factor graph SLAM:

- the odometry factor - the odometry increment tying consecutive poses,
- the observation factor - the result of the scan registration, sometime called laser odometry factors,

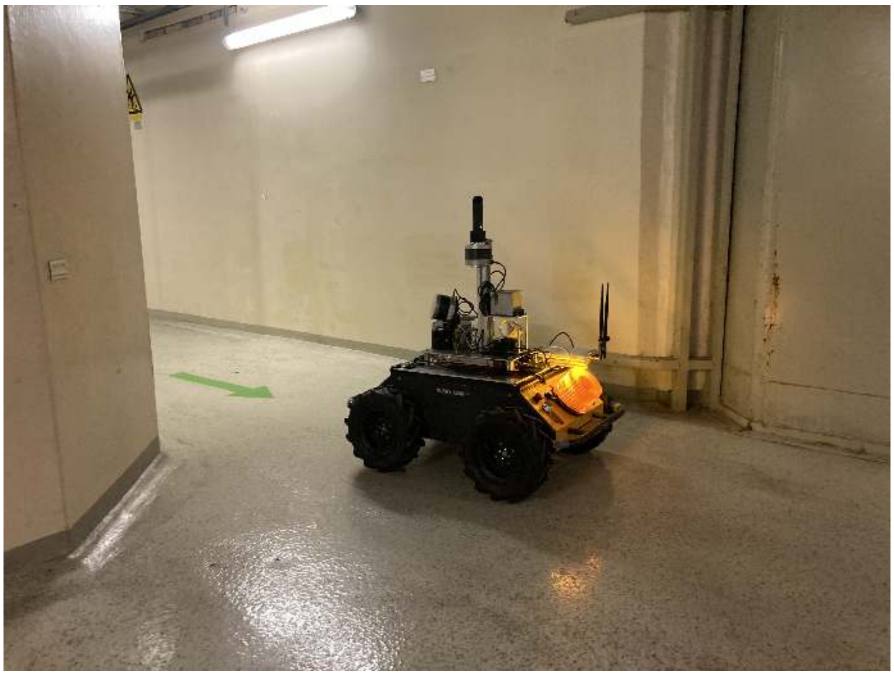


FIGURE 5.6: Robotic mapping system used for the scenario.

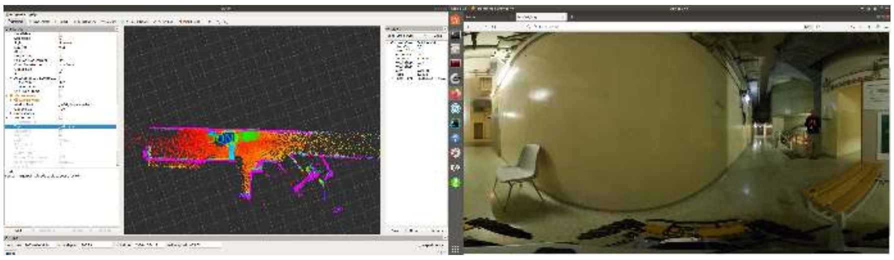


FIGURE 5.7: Screenshot from the operator's console during scenario.

- the loop-closure factor - the result of the scan registration of the loop closure candidates,
- IMU the prior factor - the factor which constrained the robot's pitch and roll,
- the optional prior factor for poses, e.g. manual scan alignment.

These are the typical factors for factor graph SLAM [113]. The structure of the factor graph is shown in figure 5.9. The factor graph approach was used because it allowed for multiple types of constraints to be introduced easily, some of which were optional. For scan registration the NDT algorithm was used. This approach is similar to a previously presented ICP point to point, but in this application it provided more robust results. The method was introduced in [81] and is often used in similar problem. The NDT algorithm decomposes registered pointcloud to a voxel grid. A 3-dimensional multivariate Gaussian distribution is fitted to the points which lie in the individual voxels. In other words, every voxel contains a mean value as the vector of a three and three by three covariance matrix. The optimization algorithm tries to refine a pose in such

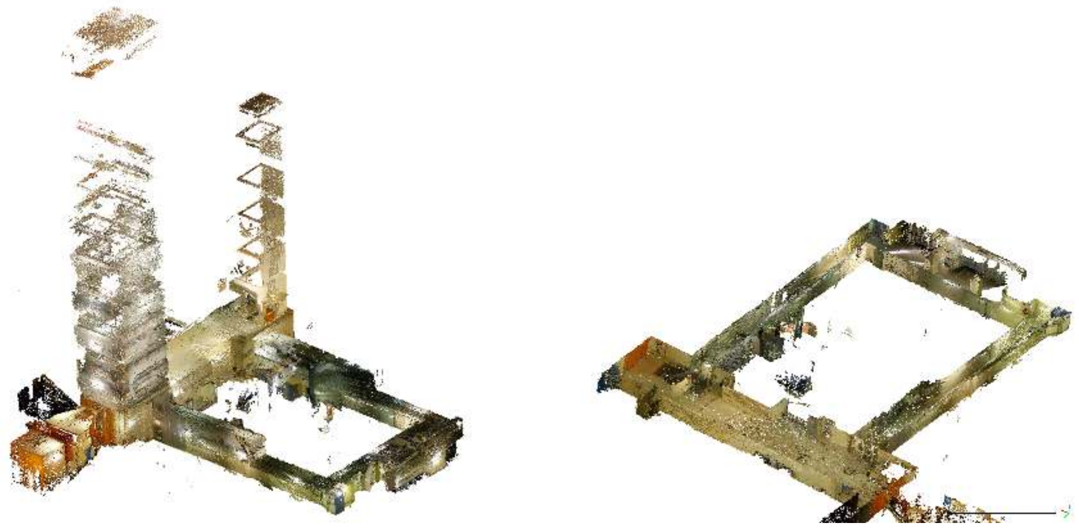


FIGURE 5.8: 3D map obtained with robotic mobile mapping system in Enrich 2021 trial with stop-scan.

a way so it maximizes the likelihood of the matched scan against the NDT voxel map. The converged NDT solution yields a relative pose which is added as a factor to factor graph. The loop closure is performed when the distance of two non-consecutive poses is smaller than the threshold. The scans which are connected in such a loop closure candidate are registered with NDT. If NDT converges, such, a candidate is added to factor graph. Finally the factor graph is optimized. A GTSAM library is used [49] for modeling and optimization. As optional refining step multiview ICP is performed. It is a time consuming step, but it improves the map consistency, especially with regards to smaller features. It is the optimization problem build from the nearest neighborhood of points in all scans. Such a nearest neighborhood contributes a point to point observation. Such large problem further optimizes the trajectory and map similarly to BA. The raw data before registration is shown in figure 5.10 where the accumulated odometry error is clearly visible in top left corner. Optimized map is shown in figure 5.11.

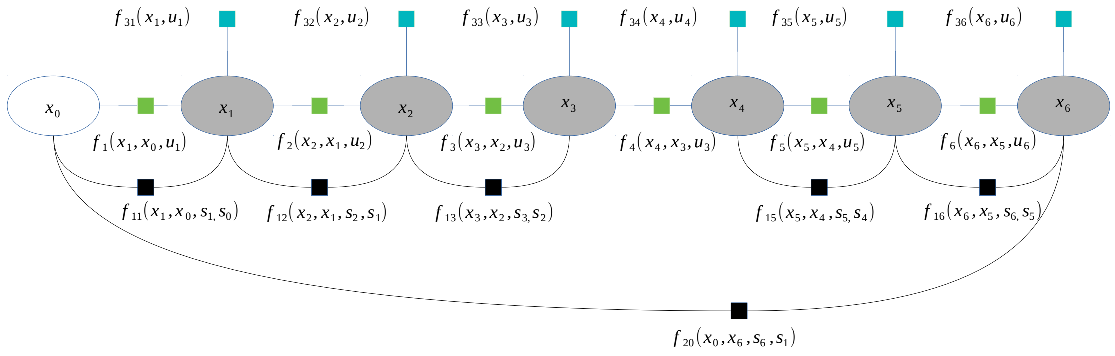


FIGURE 5.9: Used factor graph. Factors from f_1 to f_6 are odometry factors. Factors from f_{11} to f_{16} are observation factors. Factors from f_{31} to f_{36} IMU prior factors. Factors from f_{20} is a loop closure. Note that observation factor that connects pose $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ not exists, due to failed NDT matching. The variables $\boldsymbol{u}_1$ to $\boldsymbol{u}_6$ are robot odometry readings. The variables $\boldsymbol{s}_1$ to $\boldsymbol{s}_6$ are scans taken at near the corresponding poses. .

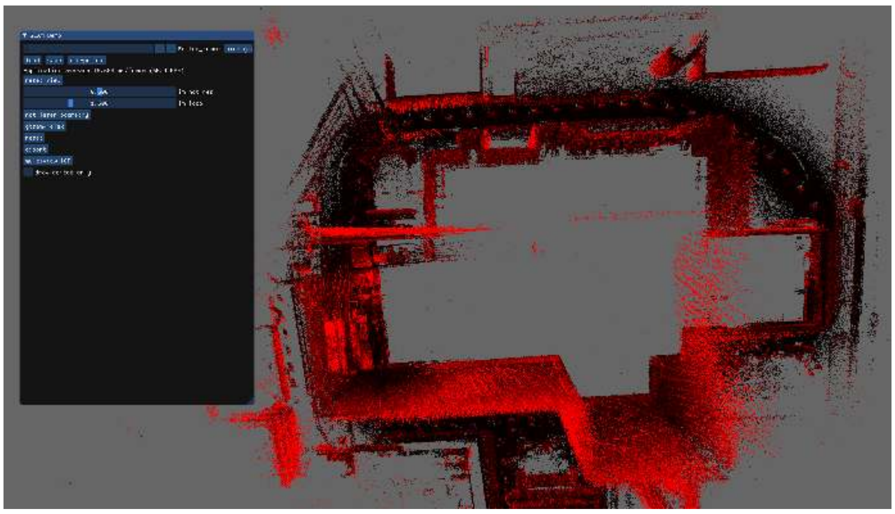


FIGURE 5.10: Tool that builds and optimize factor graph for a SLAM problem. The trajectory is not yet optimised.

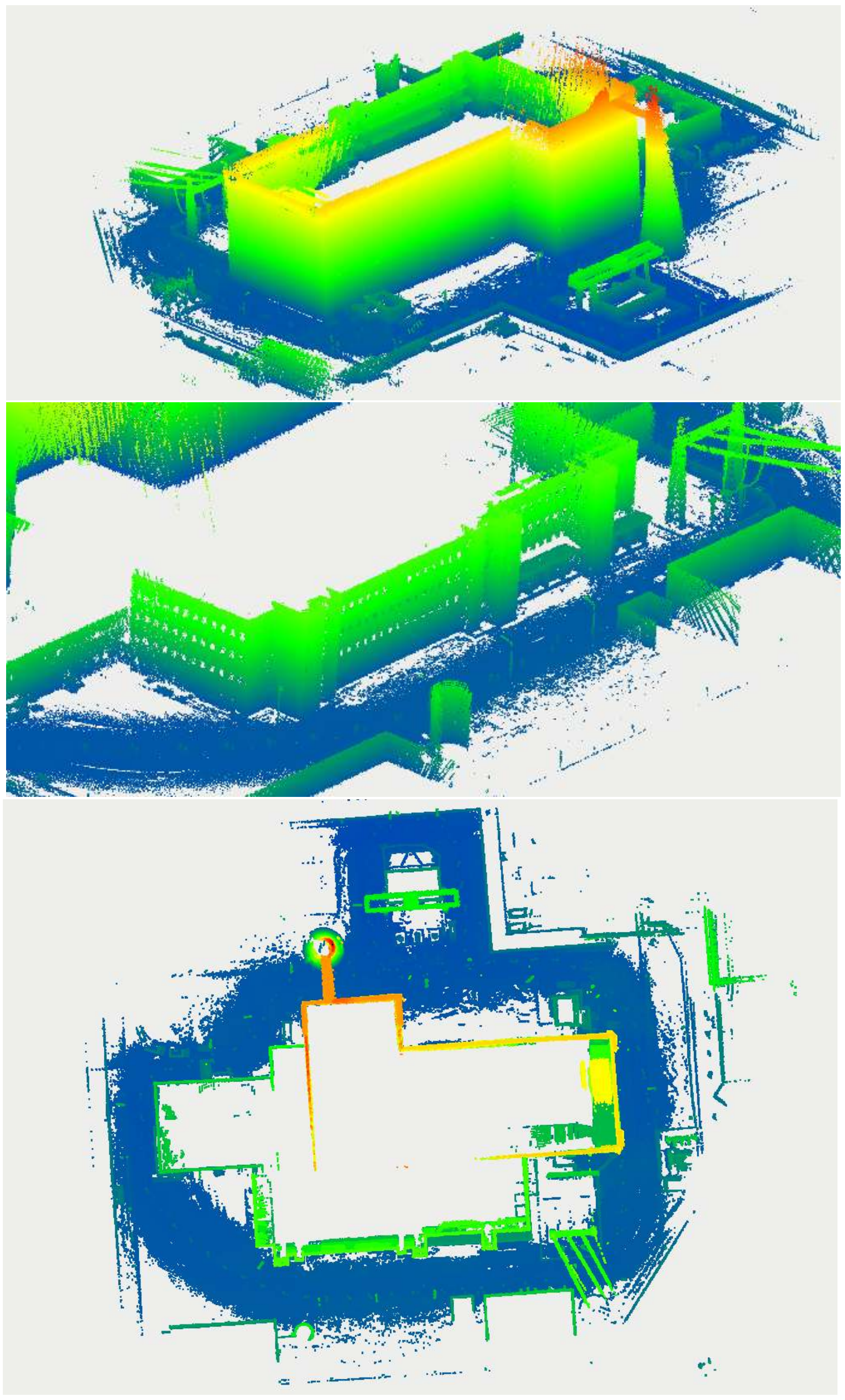


FIGURE 5.11: 3D map obtained with robotic mobile mapping system in Enrich2021 trial with mobile mapping.

5.3 Mobile robot localization

Localization using sequential Monte-Carlo

The Sequential Monte Carlo (particle filter) method was introduced in 1993 [51]. Particle filters are widely used to solve several filtering problems in signal processing. A particle filter can observe the state of a dynamic, random system with a partial and noisy observation. System noise and measurement noise do not have to be modeled by a normal distribution. An efficient localization using a particle filter is a combination of the following factors: several particles used to track state vector, a motion model, a map, and a sensor model [124]. The particle filter presented in this chapter is used in the application of localization of a mobile robot in a known and previously mapped environment. The goal of this exercise is to asses usability and precision of such system using multiple types of sensors. Three sensors solution were tested:

- a Velodyne VLP-16,
- a 3D Livox LiDAR with multi planar reflectors calibrated with method discussed in chapter 4.2,
- a 3D Livox LiDAR with multi planar reflectors with CAD derived calibration.

The robotic platform used for the test was equipped with:

- a 3D LiDAR (Velodyne VLP-16) in classical horizontal configuration,
- a 3D Livox LiDAR with planar reflectors,
- an IMU,
- a wheel odometry.

The robotic system uses an EKF for IMU and odometry fusion. Because of that, the combined odometry error is significantly reduced. The method used here, software package and its configuration are described in [87].

The examination was carried as follows: the test environment was mapped with the abovementioned robotic system. The robot was teleoperated and the data from both sensor systems was collected. The data coming from Velodyne VLP-16 and odometry were used to build a SLAM problem. Resolving SLAM results in two artifacts: a map and a trajectory. The former is used as a map for the particle filter. The latter is a ground truth trajectory for assessment of localization precision. The trajectory obtained in SLAM can be treated as the ground truth because it represents true locations of the robot during teleoperation.

Map building and trajectory ground truth

Both the Livox LiDAR and the Velodyne VLP-16 were assembled to have a common optical axis. The Velodyne LiDAR was put on top of the Livox assembly. The whole robotic system is shown in figure 5.12. As the robot was teleoperated through a given environment, data from all sensors was collected. The data from the Velodyne LiDAR was used to create a map of the test environment. The map was obtained using a factor

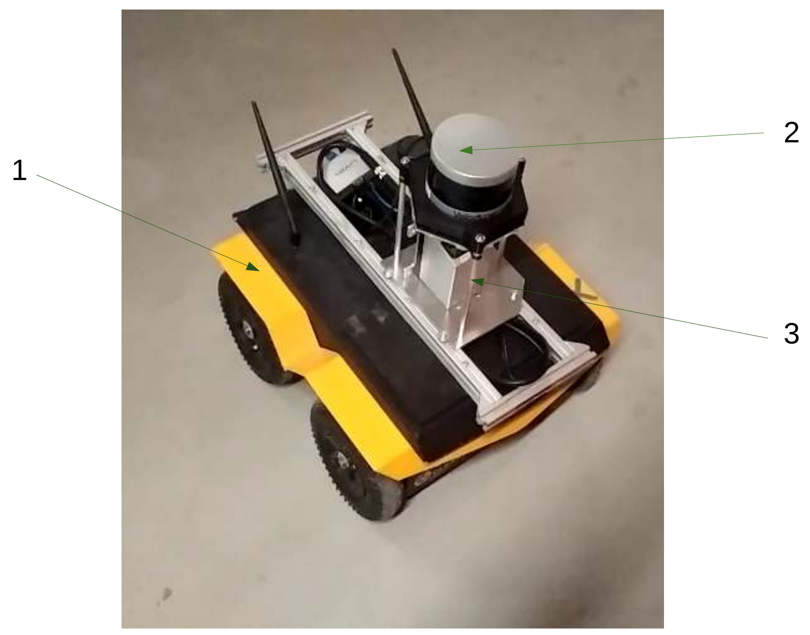


FIGURE 5.12: A robotic platform used. 1- Clearpath Jackal mobile robot, 2- Velodyne VLP-16, 3- Livox Lidar with planar reflectors.

graph SLAM introduced in a previous section. The data registration results were verified against the architectural blueprint shown in figure 5.13. The registered and optimized map yielded:

- a 3D point cloud map to be used as a representation of environment to test the proposed localization algorithm,
- a trajectory containing the robot's poses in the map's coordinate system and timestamps.

A limited number of frames was used in SLAM. Such frames are referred to as keyframes. A keyframe was added to SLAM when a distance to the last keyframe was greater than one meter. The obtained trajectory contains nodes which are approximately one meter apart. Such sparse trajectory would introduce a bias to ATE and thus cannot be used. To obtain a smooth trajectory, the second optimization problem in form of the factor graph was build. All frames, not only keyframes, were introduced as a hidden variable to a factor graph. All keyframes were fixed in place using prior factors. Those prior factors introduced poses obtained by solving SLAM problem. The noise covariance of prior factors was set to very low values. The frames between the keyframes were connected to one another with the factors of the relative pose with significantly larger noise covariance. Those factors were modeling measurements coming from robotic platform odometry. Optimizing this factor graph yielded a smooth trajectory much finer time sampling. Such trajectory could be later used to verify the performance of the localization algorithm.

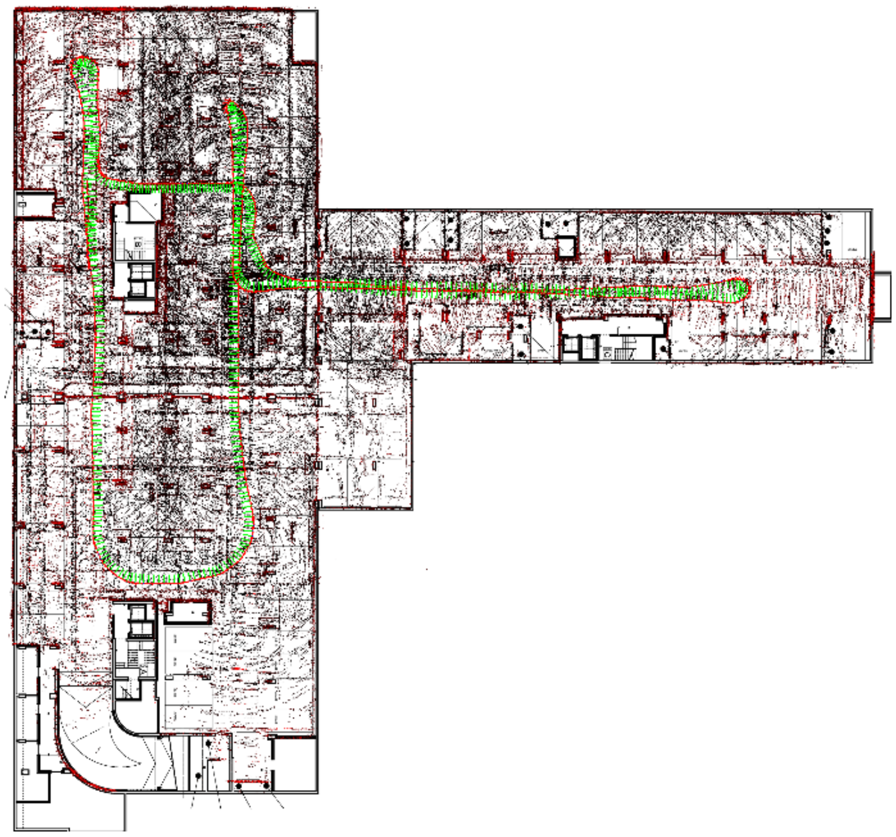


FIGURE 5.13: The result of mapping with the architectural blueprint. The trajectory is marked.

Localization algorithm

Monte Carlo Localization method represents the probability distribution of the state of the robotic system with a set of random samples. Depending on the state of the filter iteration, the target distribution is represented by the density or weight of the samples. Every sample carries the state vector hypothesis of the system in the given time step k. The localization problem is a Bayesian filtering problem, where the filter needs to find the posterior probability density of the system state given the measurements from the robot's sensor. As it is with every Bayesian filtering, the classical approach to this problem is separating the estimation into two phases: prediction and update. For the particle filter, the prediction step produces a new PDF in the form of a set of particles drawn from the last step of the algorithm. This step involves the robot's motion model. The update phase, on the other hand, incorporates the robot's sensor measurements and weighs the samples according to the sensor's model. The computed weight is used to resample. The prediction phase expects on its input a set of particles from the last iteration of the algorithm with weights updated using laser measurement. The algorithm creates a new random set where the probability of drawing particle p_i is given with its weight w_i. An algorithm called 'stochastic universal sampling' was used for this purpose [37]. The laser scanner model plays an important role in the particle filter pipeline. The model produces a simulation of a real sensor measurement for every particle. Unfortunately, it is the most computationally expensive part of the particle filter. Because modern laser scanners can produce hundreds of thousands of points per second and particle filters have to track possibly the largest number of particles, this aspect is investigated heavily in many papers [20][53]. The problem can be parallelized so that the GPGPU implementations can be found for 2D map localization, 3D map localization, and object tracking. A simplistic, robust, and effective sensor model is introduced in the implementation of the design. In the literature [124], researchers propose a sensor model (Beam-base Proximity Model) able to cover the mixture probability density of:

- measurement Gaussian noise,
- unexpected obstacle,
- random measurement,
- maximum range.

That model can be found in some widely used open-source implementations of the particle filter. AMCL implementation [106] uses this exact model, where the input is taken from the ray-traced beam and probabilities are read from the configuration file.

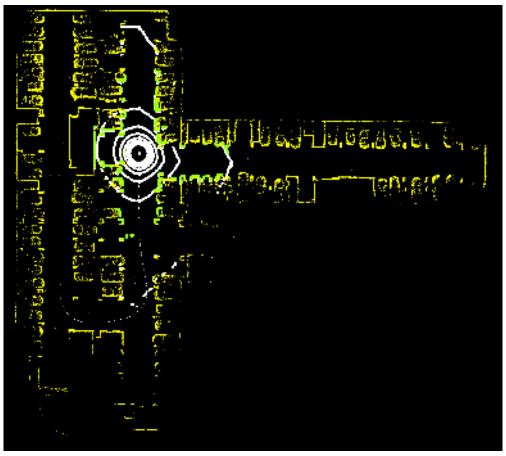
A simplified solution has been evaluated in this chapter. The map is represented as a grid of voxels with the configured dimension of the 3D cube. Those dimensions are configurable. A map is loaded from a 3D point cloud. Next, the point cloud is voxelized and stored in the GPU memory. The algorithm expects two inputs:

- an incoming laser scan from 3D laser scanner,
- a vector of 3D poses carried by particles.

The incoming point cloud is classified based on ground detection of local points height and loaded to the GPU memory. The pointcloud is decomposed into a 2D grid and for every cell the lowest point has a class of ground applied. The point which lies on the larger height is classified as obstacle. Only points with the class applied are loaded to GPU memory. Next, every 3D pose from the vector and every 3D point from the point cloud is transformed by the 3D pose and is checked if hit in an occupied voxel in a parallel manner. If so, the score counter for that pose is incremented. The pseudocode is shown in algorithm 1. The algorithm has following properties:

- the algorithm computes the overlap between sensor data and the map,
- the algorithm performs very few operations in the parallel code,
- the algorithm uses atomic GPU calls to increment the score, so there is no need to run time-consuming synchronization,
- the model is straightforward and non-physical.

The downside of this approach is the non-physical behavior of the model. The classical approach is to simulate the laser scanner measurement in a pose given by a particle. Simplified algorithm does not cover such phenomena as occlusion. Instead, it gives false-positive for map regions not visible in that pose, e.g. another side of the wall. On


(A) Visualization of the particle filter consuming data from Velodyne

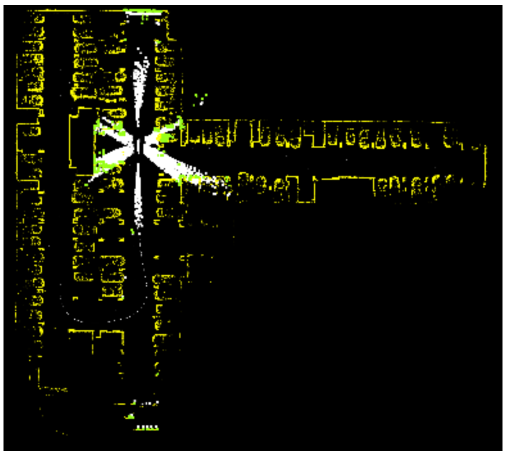

(B) Visualization of the particle filter consuming data from Livox with reflectors

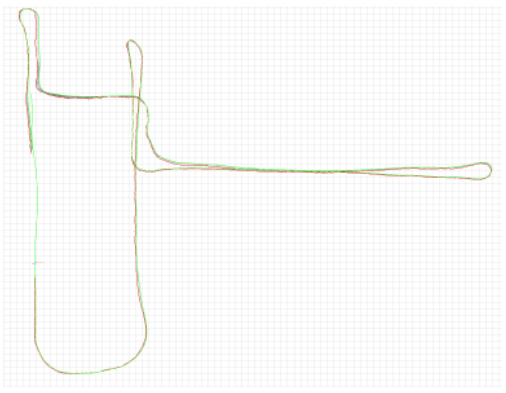
FIGURE 5.14: Comparison of input data. The yellow points are obstacles in map, the green are obstacles in incoming data from sensor, the white are ground points in data from sensor. Grey dots are positions of particles.

the other hand, this approach is very computationally efficient. The input data for both sensors systems is shown in figures 5.14a and 5.14b.

```
Algorithm 1: Simplified laser model implementation.
  Data: Voxel Map is function $M(x, y, z)$ which returns the class of a cell pointed
        by the metric coordinates $(x, y, z)$, point cloud $P = p_i$, class of i-th points
        in pointcloud is given with the $c_i$, transformations matrices which is
        carried by every particle $T = \{t_j\}$
  Result: Vector of Weights for every particle $W = w_j$
1 initialization;
2 forall $p_i$ in $P$ $\{parallel\}$ do
3     forall $t_j$ in $T$ do
4         $p_t = t_j p_i$
5         if $M(p_t) == c_i$ then
6             $w_j += 1$
7 Get $W$ from GPU memory
```

Results

The interpolated trajectory was treated as the ground truth and the localization algorithm was expected to recover the trajectory. The localization algorithm initialized after a few meters, which is faster for the Velodyne VLP-16 (fig. 5.15b) than for the Livox LiDAR with planar reflectors (fig. 5.15a). The calibrated planar reflector assembly can localize a robotic platform with precision similar to the Velodyne VLP-16. Localization algorithm can work with a various number of particles. The number of particles does not affect the error, as shown in table 5.1. A larger number of particles allows the filter to converge faster to correct solution and renders the process more robust. A smaller number of


(A) Trajectory from localization system using Livox Lidar with planar reflectors (red), ground truth trajectory (green)

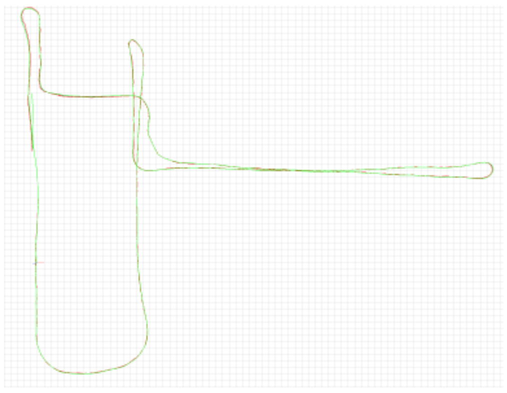

(B) Trajectory from localization system using Velodyne Lidar (red), ground truth trajectory (green)

FIGURE 5.15: Comparison of localization performance depending on input data.

samples can be used when another trustworthy method of initialization is available (e.g. beacons or tags). The calibrated system improves localization accuracy. It is caused by the fact the calibrated system is more consistent with environment in challenging places (e.g. narrow passages). Both calibrated and uncalibrated system can successfully provide input data to localization algorithm, proving robustness of presented particle filter implementation.

TABLE 5.1: ATE for multiple localization method.

Sensor type	number of particles	ATE RMSE in meters
Wheel odometry	-	13.33
Velodyne VLP-16	5 000	0.461
Velodyne VLP-16	10 000	0.407
Velodyne VLP-16	50 000	0.397
Livox with planar reflectors	5 000	0.608
Livox with planar reflectors	10 000	0.611
Livox with planar reflectors	50 000	0.614
Livox with planar reflectors, uncalibrated	5 000	0.664
Livox with planar reflectors, uncalibrated	10 000	0.664
Livox with planar reflectors, uncalibrated	50 000	0.667

Chapter 6

Conclusions

The main thesis of the dissertation has been proven true since the automation of the calibration process of a multi-sensor robotic mobile mapping system reduces the amount of expert knowledge required within the context of autonomous mobile robots performing tasks in unknown environments and increase the autonomy of the system. The expert knowledge required in the typical calibration procedure is that of the metrological properties of the environment and other similarly important information required for achieving accurate measurements. The proposed methodology is discussed in chapter 3. It emphasizes the importance of time synchronization in section 3.1, the representation of the rotation matrix and the motion model discussed in section 3.2. The approach to SLAM was discussed in section 3.3. It introduces automatic tools whose purpose is to improve the calibration procedure by incorporating LiDAR and camera observation equations discussed in sections 3.4 and 3.5. Moreover, the proposed calibration procedure can utilize the additional markers or reference data source. As shown in chapter 4, automatic calibration can be also performed using the multi view data registration scheme. Furthermore, such calibration procedure is proposed for multi-sensor systems where the overlap between sensors is insufficient, e.g. targeted autonomous mobile robot applications whose mission is performed in an unknown environment. Thus, data collected by a mobile robot can be used for calibration.

Since the possible mechanical issues are addressed, the thesis contributes to improvement of the autonomous mobile robot system deployment and long-term exploitation. Mechanical problems can occur during the transportation of the system as it was observed in the practical application discussed in section 4.1. For instance, the high-volume mobile 3D scanner is composed of joints which have to be dismantled for each deployment. Other potential mechanical difficulties are discussed in section 5.1 where the mining shaft mapping application is detailed. In such a case, extreme environmental conditions require the system to be re-calibrated for each measurement because of mechanical shocks which could occur due to the impact with unknown obstacles. This particular application serves as a demonstration for the fact that a typical calibration procedure performed in laboratory can not be universally applied since the measurement instrument is de-mounted for transportation and mounted on the site. Moreover, the probability of impact with unknown obstacles is high; therefore, the extrinsic calibration can change over time with exploitation of the system. The proposed method can be extended for mobile mapping application such as nuclear power plant mapping (as discussed in section 5.2). The direct positive impact into mobile robotics domain verified as mobile robot localization, in section 5.3, shows that the proposed methodology can be incorporated in various mobile robotic applications. The research fulfills the gap in recent mobile robotics studies by showing the importance of data synchronization and automation of the calibration process for obtaining highly accurate mobile mapping results.

6.1 Thesis contributions

The main contribution of the thesis is the methodology for the automation of the calibration process of the multi-sensor robotic mobile mapping system. The thesis demonstrates the capability to increase the autonomy of the mobile robot performing the task in an unknown environment based on many real world experiments. The direct impact is the automation of the calibration procedure resulting in the reduction of required expert knowledge. This work improves the deployment of the mobile robotic applications in real world conditions. Four more precise theses supporting the main thesis have been confirmed as follows:

- automation of the calibration process reduces the need for expert knowledge required for accurate measurement of intrinsic and extrinsic calibration parameters (section Calibration of a high-volume mobile 3D scanner page 63, section Calibration of multi planar reflectors page 73, section Calibration of a rotated reflector page 83, section Calibration of a lightweight 3D unit page 89, section Calibration of a mobile backpack mapping system page 97, section Calibration of a 3D unit page 107 and section Mining shaft mapping page 113),
- the new method for reshaping the field of view of modern Solid State LiDARs enables customizing the robotic mobile mapping systems for various applications (section Calibration of multi planar reflectors page 73 and section Calibration of a rotated reflector page 83),
- the chosen rotation matrix parametrization enables robust optimization of the calibration parameters (all solutions mentioned in chapter Experimental validation page 63 and chapter Robotic applications page 113),
- automatic calibration enables long term autonomous mobile robot inspection of the unknown environment by reducing mechanical issues related to the robot's exploitation (section Mining shaft mapping page 113).

Other relevant contributions are:

- the methodology for automation of the multi-sensor system calibration for mobile robotic applications (section Methodology page 20),
- the open source projects related with multi planar [97] and rotated reflectors [96],
- the design and research of a high-volume mobile 3D scanner (section Calibration of a high-volume mobile 3D scanner page 63),
- the design and research of a new 3D sensor for mobile robotic applications (section Calibration of multi planar reflectors page 4.2 and section Calibration of a rotated reflector page 83),

- the design and research of a lightweight 3D unit (section Calibration of a lightweight 3D unit page 89),
- the design and research of a mobile backpack mapping system (section Calibration of a mobile backpack mapping system and 97),
- the design and research of a 3D unit composed of rotated LiDAR and a spherical camera (section 4.6),
- the design and deployment of a mining shaft mapping system (section Mining shaft mapping page 113),
- the evaluation of proposed methodology in various realistic conditions, including nuclear power plant mapping, and navigation (section Nuclear Power Plant Mapping page 120 and section Mobile robot localization page 125).

The implementation of the automatic calibration is scalable; thus, it can be used for designing, developing and deploying other multi-sensor systems.

6.2 Further research directions

Implementation, development and deployment of mobile robotic systems are demanding tasks. For this reason, the proposed methodology will be extended by other automation techniques which allow for decreasing of the need for human operations. Possible interesting research direction is the use of AI techniques enabling image and point-cloud interpretation. Such approach could improve calibration pattern recognition.

Bibliography

[1] Artur Adamek and Janusz Będkowski. *Automated Mobile System for Mapping Mine Shafts*. https://www.gim-international.com/content/article/automated-mobile-system-for-mapping-mine-shafts. [Online; accessed 31st March 2022].

[2] Sameer Agarwal, Keir Mierle, and Others. *Ceres Solver*. http://ceres-solver.org. [Online; accessed 31st March 2022].

[3] *Apollo 11 Air-to-ground voice transcription*. https://www.hq.nasa.gov/alsj/a11/a11transcript_tec.html. Page 337 [Online; accessed 31st March 2022].

[4] Minoru Asada and Yoshiaki Shirai. "Building a world model for a mobile robot using dynamic semantic constraints". In: *Proc. 11 th International Joint Conference on Artificial Intelligence*. 1989, pp. 1629–1634.

[5] *Automated Vehicles Require Intelligent HD Maps for Reliable and Safe Driving*. https://nds-association.org/hd-maps/. [Online; accessed 31st March 2022].

[6] Jonathan T. Barron. "A More General Robust Loss Function". In: *CoRR* abs/1701.03077 (2017). arXiv: 1701.03077. URL: http://arxiv.org/abs/1701.03077.

[7] A. Bartoli and P. Sturm. "The 3D line motion matrix and alignment of line reconstructions". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990488.

[8] Dominik Belter, Michał Nowicki, and Piotr Skrzypczyński. "On the Performance of Pose-based RGB-D Visual Navigation Systems". In: vol. 9004. Apr. 2015, pp. 407–423. ISBN: 9783319168074. DOI: 10.1007/978-3-319-16808-1_28.

[9] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[10] Felix Bettonvil. "Fisheye lenses". In: *WGN, Journal of the International Meteor Organization* 33 (Jan. 2005), pp. 9–14.

[11] Jose-Luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. 012010. University of Malaga, 2010. URL: http://ingmec.ual.es/~jlblanco/papers/jlblanco2010geometry3D\%5Ftechrep.pdf.

[12] Jose Luis Blanco and Pranjal Kumar Rai. *nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees*. https://github.com/jlblancoc/nanoflann. [Online; accessed 31st March 2022]. 2014.

[13] *Blender : The Freedom to Create*. https://www.blender.org/about/. [Online; accessed 31st March 2022].

[14] *Blender Market Classical Library for Blender Eevee and Cycles*. https://blendermarket.com/products/classic-library-for-blender-eevee-and-cycles. [Online; accessed 31st March 2022].

[15] Fabian Blöchliger et al. "Topomap: Topological Mapping and Navigation Based on Visual SLAM Maps". In: *CoRR* abs/1709.05533 (2017). arXiv: 1709.05533. URL: http://arxiv.org/abs/1709.05533.

[16] Ezekiel Bokolonga et al. "A compact multispectral image capture unit for deployment on drones". In: May 2016, pp. 1–5. DOI: 10.1109/i2mtc.2016.7520445.

[17] Michael J Broxton and Larry J Edwards. "The Ames Stereo Pipeline: Automated 3D surface reconstruction from orbital imagery". In: *Lunar and planetary science conference*. 1391. 2008, p. 2419.

[18] Rafael Y Brzezinski et al. "Automated processing of thermal imaging to detect COVID-19". In: *Scientific Reports* 11.1 (Sept. 2021), p. 17489.

[19] D. J. Butler et al. "A naturalistic open source movie for optical flow evaluation". In: *European Conf. on Computer Vision (ECCV)*. Ed. by A. Fitzgibbon et al. (Eds.) Part IV, LNCS 7577. Springer-Verlag, Oct. 2012, pp. 611–625.

[20] Janusz Będkowski and Timo Röhling. "Online 3D LIDAR Monte Carlo localization with GPU acceleration". In: *Industrial Robot: An International Journal* 44.4 (2017), pp. 442–456.

[21] Janusz Będkowski et al. "A Novel Approach to Global Positioning System Accuracy Assessment, Verified on LiDAR Alignment of One Million Kilometers at a Continent Scale, as a Foundation for Autonomous DRIVING Safety Analysis". In: *Sensors* 21.17 (2021). ISSN: 1424-8220. DOI: 10.3390/s21175691. URL: https://www.mdpi.com/1424-8220/21/17/5691.

[22] Yohann Cabon, Naila Murray, and Martin Humenberger. *Virtual KITTI 2*. 2020. arXiv: 2001.10773 [cs.CV].

[23] H. Cantzler, Robert B. Fisher, and Michel Devy. "Quality Enhancement of Reconstructed 3D Models Using Coplanarity and Constraints". In: *Proceedings of the 24th DAGM Symposium on Pattern Recognition*. London, UK: Springer-Verlag, 2002, pp. 34–41. ISBN: 3-540-44209-x.

[24] Luca Carlone and Andrea Censi. "From Angular Manifolds to the Integer Lattice: Guaranteed Orientation Estimation With Application to Pose Graph Optimization". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 475–492. DOI: 10.1109/tro.2013.2291626.

[25] João Cartucho et al. "VisionBlender: a tool to efficiently generate computer vision datasets for robotic surgery". In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* (2020), pp. 1–8.

[26] Nived Chebrolu et al. "Adaptive Robust Kernels for Non-Linear Least Squares Problems". In: *CoRR* abs/2004.14938 (2020). arXiv: 2004.14938. URL: https://arxiv.org/abs/2004.14938.

[27] Nived Chebrolu et al. *Adaptive Robust Kernels for Non-Linear Least Squares Problems*. 2021. arXiv: 2004.14938 [cs.RO].

[28] Yu Chen, Yisong Chen, and Guoping Wang. *Bundle Adjustment Revisited*. 2019. arXiv: 1912.03858 [cs.CV].

[29] D. Chulkovs, E. Grabs, and A. Ipatovs. “Comparison of MEMS and FOG Gyroscopes for Daily Use in Camera Stabilizing Systems”. In: *2020 24th International Conference Electronics*. 2020, pp. 1–4. DOI: 10.1109/ieeeconf49502.2020.9141619.

[30] Hai fa Dai et al. “An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network”. In: *Defence Technology* 16.2 (2020), pp. 334–340. ISSN: 2214-9147. DOI: https://doi.org/10.1016/j.dt.2019.08.011. URL: https://www.sciencedirect.com/science/article/pii/S2214914719303058.

[31] Thao Dang, Christian Hoffmann, and Christoph Stiller. “Continuous Stereo Self-Calibration by Camera Parameter Tracking”. In: *IEEE Transactions on Image Processing* 18.7 (2009), pp. 1536–1550. DOI: 10.1109/tip.2009.2017824.

[32] Povilas Daniušis et al. “Topological navigation graph framework”. In: *Autonomous Robots* 45.5 (2021), pp. 633–646. ISSN: 1573-7527. DOI: 10.1007/s10514-021-09980-x. URL: https://doi.org/10.1007/s10514-021-09980-x.

[33] Frank Dellaert and Michael Kaess. “Factor Graphs for Robot Perception”. In: *Found. Trends Robotics* 6 (2017), pp. 1–139.

[34] Olaf Deppe et al. “MEMS and FOG Technologies for Tactical and Navigation Grade Inertial Sensors–Recent Improvements and Comparison”. In: *Sensors* 17.3 (2017). ISSN: 1424-8220. DOI: 10.3390/s17030567. URL: https://www.mdpi.com/1424-8220/17/3/567.

[35] M.W.M. Dissanayake et al. “A Solution to the Simultaneous Localization and Map Building (SLAM) Problem”. In: *Robotics and Automation, IEEE Transactions on* 17 (July 2001), pp. 229–241. DOI: 10.1109/70.938381.

[36] Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator*. 2017. arXiv: 1711.03938 [cs.LG].

[37] R. Douc and O. Cappe. “Comparison of resampling schemes for particle filtering”. In: *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* 2005, pp. 64–69. DOI: 10.1109/ispa.2005.195385.

[38] Mamadou Doumbia and Xu Cheng. “State Estimation and Localization Based on Sensor Fusion for Autonomous Robots in Indoor Environment”. In: *Computers* 9.4 (2020). ISSN: 2073-431x. DOI: 10.3390/computers9040084. URL: https://www.mdpi.com/2073-431X/9/4/84.

[39] S.P.H. Driessen et al. “Experimentally Validated Extended Kalman Filter for UAV State Estimation Using Low-Cost Sensors”. In: *IFAC-PapersOnLine* 51.15 (2018). 18th IFAC Symposium on System Identification SYSID 2018, pp. 43–48. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.09.088. URL: https://www.sciencedirect.com/science/article/pii/S2405896318317488.

[40] Hao Du et al. “Real-Time Onboard 3D State Estimation of an Unmanned Aerial Vehicle in Multi-Environments Using Multi-Sensor Data Fusion”. In: *Sensors* 20.3 (2020). ISSN: 1424-8220. DOI: 10.3390/s20030919. URL: https://www.mdpi.com/1424-8220/20/3/919.

[41] Yanming Feng and Jinling Wang. “GPS RTK Performance Characteristics and Analysis”. In: *Journal of Global Positioning Systems* 7 (June 2008). DOI: 10.5081/jgps.7.1.1.

[42] M. A. Fischler and R. Bolles. “Random sample consensus. A paradigm for model fitting with apphcahons to image analysm and automated cartography”. In: *Proc. 1980 Image Understandtng Workshop (College Park, Md., Apr i980) L. S. Baurnann, Ed, Scmnce Apphcatlons, McLean, Va.* 1980, pp. 71–88.

[43] Tully Foote. “tf: The transform library”. In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. 2013, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

[44] U. Frese. “A Proof for the Approximate Sparsity of SLAM Information Matrices”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. 2005, pp. 329–335. DOI: 10.1109/robot.2005.1570140.

[45] Xiang Gao et al. *14 Lectures on Visual SLAM: From Theory to Practice*. Publishing House of Electronics Industry, 2017.

[46] Sergio Garrido-Jurado et al. “Generation of fiducial marker dictionaries using Mixed Integer Linear Programming”. In: *Pattern Recognition* 51 (Oct. 2015). DOI: 10.1016/j.patcog.2015.09.023.

[47] Deepak Gautam et al. “Comparison of MEMS-Based and FOG-Based IMUs to Determine Sensor Pose on an Unmanned Aircraft System”. In: *Journal of Surveying Engineering* 143.4 (2017), p. 04017009. DOI: 10.1061/(asce)su.1943-5428.0000225. eprint: https://ascelibrary.org/doi/abs/10.1061/(ASCE)SU.1943-5428.0000225. URL: https://ascelibrary.org/doi/abs/10.1061/(ASCE)SU.1943-5428.0000225.

[48] Henri P. Gavin. “The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems”. In: Department of Civil and Environmental Engineering, Duke University, 2013, pp. 1,19.

[49] *Georgia Tech smoothing and mapping library*. https://gtsam.org. [Online; accessed 31st March 2022].

[50] C. Goodall et al. “The Battle Between MEMS and FOGs for Precision Guidance”. In: [Online; accessed 31st March 2022]. 2013.

[51] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEE Proceedings F - Radar and Signal Processing* 140.2 (1993), pp. 107–113. ISSN: 0956-375x. DOI: 10.1049/ip-f-2.1993.0015.

[52] Caterina Gottardi and Francesco Guerra. “Spherical Images For Cultural Heritage: Survey And Documentation With The Nikon Km360”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* Xlii-2 (May 2018), pp. 385–390. DOI: 10.5194/isprs-archives-XLII-2-385-2018.

[53] Bhavya Goyal et al. “Implementation of Particle Filters for Single Target Tracking Using CUDA”. In: Sept. 2015, pp. 28–32. DOI: 10.1109/icacc.2015.111.

[54] O. Grau. “A scene analysis system for the generation of 3-D models”. In: *NRC ’97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*. Washington, DC, USA: IEEE Computer Society, 1997, p. 221. ISBN: 0-8186-7943-3.

[55] Giorgio Grisetti et al. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. DOI: 10.1109/mits.2010.939925.

[56] Carlos San Vicente Gutiérrez et al. “Time Synchronization in modular collaborative robots”. In: *CoRR* abs/1809.07295 (2018). arXiv: 1809.07295. URL: http://arxiv.org/abs/1809.07295.

[57] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[58] Bernd Heidtmann. *GNSS receiver accuracy: Closing the gap between the datasheet and reality*. https://www.u-blox.com/en/blogs/tech/gnss-receiver-accuracy-closing-gap-between-datasheet-and-reality. [Online; accessed 31st March 2022].

[59] *HERE HD Live Map Technical Paper*. https://engage.here.com/hubfs/Downloads/Tech%20Briefs/HERE%20Technologies%20Self-healing%20Map%20Tech%20Brief.pdf. [Online; accessed 31st March 2022].

[60] Seon Ho Oh. and Soon Ki Jung. “A great circle arc detector in equirectangular images”. In: *Proceedings of the International Conference on Computer Vision Theory and Applications - Volume 2: VISAPP, (VISIGRAPP 2012)*. INSTICC. SciTePress, 2012, pp. 346–351. ISBN: 978-989-8565-03-7. DOI: 10.5220/0003856303460351.

[61] Kai Hormann and Alexander Agathos. “The point in polygon problem for arbitrary polygons”. In: *Computational Geometry* 20.3 (2001), pp. 131–144. ISSN: 0925-7721. DOI: https://doi.org/10.1016/S0925-7721(01)00012-8. URL: https://www.sciencedirect.com/science/article/pii/S0925772101000128.

[62] Berthold Horn, Hugh Hilden, and Shahriar Negahdaripour. “Closed-Form Solution of Absolute Orientation using Orthonormal Matrices”. In: *Journal of the Optical Society of America A* 5 (July 1988), pp. 1127–1135. DOI: 10.1364/josaa.5.001127.

[63] Qiqi Hou et al. *Fast Monte Carlo Rendering via Multi-Resolution Sampling*. 2021. DOI: 10.48550/ARXIV.2106.12802. URL: https://arxiv.org/abs/2106.12802.

[64] *How we build reality Z+F IMAGER® 5010*. https://www.hts-3d.com/techSheets/Z+F-IMAGER-5010-tech-sheet.pdf. [Online; accessed 31st March 2022].

[65] Yinghao Huang et al. "Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 37 (Nov. 2018). Two first authors contributed equally, 185:1–185:15. DOI: https://doi.org/10.1145/3272127.3275108.

[66] *Intel® RealSense™ Documentation*. https://dev.intelrealsense.com/docs. [Online; accessed 31st March 2022].

[67] Shunping Ji et al. "Comparison of Two Panoramic Sensor Models for Precise 3D Measurements". In: *Photogrammetric Engineering & Remote Sensing* 80 (Mar. 2014), pp. 229–238. DOI: 10.14358/pers.80.3.229.

[68] Malek Karaim, Mohamed Elsheikh, and Aboelmagd Noureldin. "GNSS Error Sources". In: *Multifunctional Operation and Application of GPS*. Ed. by Rustam B. Rustamov and Arif M. Hashimov. Rijeka: IntechOpen, 2018. Chap. 4. DOI: 10.5772/intechopen.75493. URL: https://doi.org/10.5772/intechopen.75493.

[69] Faisal Khan et al. *High-Accuracy Facial Depth Models derived from 3D Synthetic Data*. 2020. arXiv: 2003.06211 [eess.IV].

[70] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/iros.2004.1389727.

[71] Y. N. Korkishko et al. "High-precision inertial measurement unit IMU-5000". In: *2018 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*. 2018, pp. 1–4. DOI: 10.1109/isiss.2018.8358121.

[72] David Kortenkamp and Terry Weymouth. "Topological Mapping for Mobile Robots Using a Combination of Sonar and Vision Sensing". In: *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*. Aaai'94. Seattle, Washington, USA: American Association for Artificial Intelligence, 1994, 979–984. ISBN: 0262611023.

[73] Rainer Kümmerle et al. "g2o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3607–3613. DOI: 10.1109/icra.2011.5979949.

[74] Karol Kwiatek. "Imersyjny system mobilny do fotogrametrycznych pomiarów 3D". In: *Imersyjny system mobilny do fotogrametrycznych pomiarów 3D*. Ed. by Monika Filipek. Krakow: Wydawnicta Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, 2020. ISBN: 978-83-66016-92-7.

[75] Herbert Landau et al. "Trimble's RTK And DGPS Solutions In Comparison With Precise Point Positioning". In: *Observing our Changing Earth*. Ed. by Michael G. Sideris. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 709–718. ISBN: 978-3-540-85426-5.

[76] Thomas Lemaire and Simon Lacroix. "Monocular-vision based SLAM using Line Segments". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 2791–2796. DOI: 10.1109/ROBOT.2007.363894.

[77] Zheng Liu, Fu Zhang, and Xiaoping Hong. *Low-cost Retina-like Robotic Lidars Based on Incommensurable Scanning*. 2020. arXiv: 2006.11034 [cs.RO].

[78] Zheng Liu, Fu Zhang, and Xiaoping Hong. "Low-cost Retina-like Robotic Lidars Based on Incommensurable Scanning". In: *CoRR* abs/2006.11034 (2020). arXiv: 2006.11034. URL: https://arxiv.org/abs/2006.11034.

[79] *Livox Mid-series*. https://www.livoxtech.com/mid-40-and-mid-100. [Online; accessed 31st March 2022].

[80] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. "Methods for Non-Linear Least Squares Problems (2nd ed.)" In: 2004.

[81] Martin Magnusson. "The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection". PhD thesis. Dec. 2009.

[82] Agostino Martinelli et al. "Simultaneous localization and odometry calibration for mobile robot". In: *Iros 2003*. Vol. 2. Las Vegas, United States, Oct. 2003, pp. 1499–1504. DOI: 10.1109/iros.2003.1248856. URL: https://hal.archives-ouvertes.fr/hal-01015907.

[83] N. Mayer et al. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:1512.02134. 2016. URL: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16.

[84] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[85] Vu Trieu Minh, Nikita Katushin, and John Pumwa. "Motion tracking glove for augmented reality and virtual reality:" in: *Paladyn, Journal of Behavioral Robotics* 10.1 (2019), pp. 160–166. DOI: doi:10.1515/pjbr-2019-0012. URL: https://doi.org/10.1515/pjbr-2019-0012.

[86] Michael Montemerlo et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: Nov. 2002.

[87] T. Moore and D. Stouch. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, 2014.

[88] M. Moskowitz. *A Course in Complex Analysis in One Variable*. World Scientific, 2002. DOI: 10.1142/4836.

[89] *MTi User Manual*. [Online; accessed 31st March 2022]. URL: https://www.xsens.com/hubfs/Downloads/usermanual/MTi_usermanual.pdf.

[90] Marion Mundt et al. "Estimation of Gait Mechanics Based on Simulated and Measured IMU Data Using an Artificial Neural Network". In: *Frontiers in Bioengineering and Biotechnology* 8 (2020), p. 41. ISSN: 2296-4185. DOI: 10.3389/fbioe.2020.00041. URL: https://www.frontiersin.org/article/10.3389/fbioe.2020.00041.

[91] Tae Hyeon Nam, Jae Hong Shim, and Young Im Cho. "A 2.5D Map-Based Mobile Robot Localization via Cooperation of Aerial and Ground Robots". In: *Sensors* 17.12 (2017). ISSN: 1424-8220. DOI: 10.3390/s17122730. URL: https://www.mdpi.com/1424-8220/17/12/2730.

[92] Andreas Nüchter and Joachim Hertzberg. "Towards semantic maps for mobile robots". In: *Robot. Auton. Syst.* 56.11 (2008), pp. 915–926. ISSN: 0921-8890.

[93] Andreas Nüchter et al. "3D Mapping with Semantic Knowledge". In: *In Robocup International Symposium*. 2005, pp. 335–346.

[94] Andreas Nüchter et al. "Semantic Scene Analysis of Scanned 3D Indoor Environments". In: *in: Proceedings of the Eighth International Fall Workshop on Vision, Modeling, and Visualization (VMV 03)*. 2003, pp. 658–666.

[95] Vittorio M. N. Passaro et al. "Gyroscope Technology and Applications: A Review in the Industrial Perspective". In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: 10.3390/s17102284. URL: https://www.mdpi.com/1424-8220/17/10/2284.

[96] Michal Pełka. *Lidar spinning Mirror*. https://github.com/michalpelka/lidar_spinning_mirror. [Online; accessed 31st March 2022].

[97] Michał Pełka and Janusz Będkowski. "Calibration of Planar Reflectors Reshaping LiDAR's Field of View". In: *Sensors* 21.19 (2021). ISSN: 1424-8220. DOI: 10.3390/s21196501. URL: https://www.mdpi.com/1424-8220/21/19/6501.

[98] S. Phillips et al. "A Careful Consideration of the Calibration Concept". In: *Journal of research of the National Institute of Standards and Technology* 106,2 371-9 (2001). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4862813/.

[99] Fabian Poggenhans et al. "Lanelet2: A High-Definition Map Framework for the Future of Automated Driving". In: *Proc. IEEE Intell. Trans. Syst. Conf.* Hawaii, USA, 2018. URL: http://www.mrt.kit.edu/z/publ/download/2018/Poggenhans2018Lanelet2.pdf.

[100] Mr Prasad, Tadikonda Srinivasulu, and Ashok Yadav. "Embedded MEMS: A New Era in Mobile Technology". In: 3 (Mar. 2013), pp. 857–862.

[101] David Prokhorov et al. "Measuring robustness of Visual SLAM". In: *2019 16th International Conference on Machine Vision Applications (MVA)*. 2019, pp. 1–6. DOI: 10.23919/MVA.2019.8758020.

[102] Albert Pumarola et al. "PL-SLAM: Real-time monocular visual SLAM with points and lines". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4503–4508. DOI: 10.1109/ICRA.2017.7989522.

[103] Sheng Qinghong et al. "Registration of Aerial Image with Airborne LiDAR Data Based on Plücker Line". In: *Acta Geodaetica et Cartographica Sinica* 44.7, 761 (2015), p. 761. DOI: 10.11947/j.AGCS.2015.20140123. URL: http://xb.sinomaps.com/EN/abstract/article\%5F6575.shtml.

[104] Weichao Qiu and Alan L. Yuille. "UnrealCV: Connecting Computer Vision to Unreal Engine". In: *CoRR* abs/1609.01326 (2016). arXiv: 1609.01326. URL: http://arxiv.org/abs/1609.01326.

[105] Francisco Romero-Ramirez, Rafael Muñoz Salinas, and Rafael Medina-Carnicer. "Speeded Up Detection of Squared Fiducial Markers". In: *Image and Vision Computing* 76 (June 2018). DOI: 10.1016/j.imavis.2018.05.004.

[106] *ROS Planning/navigation*. https://github.com/ros-planning/navigation. [Online; accessed 31st March 2022].

[107] Marta Rostkowska and Piotr Skrzypczyński. "Hybrid field of view vision: From biological inspirations to integrated sensor design". In: *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2016, pp. 629–634. DOI: 10.1109/MFI.2016.7849557.

[108] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4. DOI: 10.1109/icra.2011.5980567.

[109] Joaquim Salvi, Xavier Armangué, and Joan Batlle. "A comparative review of camera calibrating methods with accuracy evaluation". In: *Pattern Recognition* 35.7 (2002), pp. 1617–1635. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(01)00126-1. URL: https://www.sciencedirect.com/science/article/pii/S0031320301001261.

[110] Marcus Schmidt et al. "IMU- based Determination of Stance Duration During Sprinting". In: *Procedia Engineering* 147 (2016). The Engineering of SPORT 11, pp. 747–752. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2016.06.330. URL: https://www.sciencedirect.com/science/article/pii/S1877705816307779.

[111] Thomas Schneider et al. "Visual-inertial self-calibration on informative motion segments". In: May 2017, pp. 6487–6494. DOI: 10.1109/icra.2017.7989766.

[112] Gamal Seedahmed and Anton Schenk. "Direct Linear Transformation In The Context Of Different Scaling Criteria". In: Jan. 2001. DOI: 10.13140/2.1.3600.4644.

[113] Tixiao Shan et al. "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Ieee. 2020, pp. 5135–5142.

[114] Ken Shoemake. "Animating Rotation with Quaternion Curves". In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), 245–254. ISSN: 0097-8930. DOI: 10.1145/325165.325242. URL: https://doi.org/10.1145/325165.325242.

[115] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. Tech. rep. 2018. arXiv: 1812.01537. URL: http://arxiv.org/abs/1812.01537.

[116] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 2012.

[117] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. “OpenVSLAM: A Versatile Visual SLAM Framework”. In: *CoRR* abs/1910.01122 (2019). arXiv: `1910.01122`. URL: `http://arxiv.org/abs/1910.01122`.

[118] Zheng Sun and Yingying Zhang. “Accuracy Evaluation of Videogrammetry Using A Low-Cost Spherical Camera for Narrow Architectural Heritage: An Observational Study with Variable Baselines and Blur Filters”. In: *Sensors* 19 (Jan. 2019), p. 496. DOI: `10.3390/s19030496`.

[119] A. Szajewska. “Development of the Thermal Imaging Camera (TIC) Technology”. In: *Procedia Engineering* 172 (2017). Modern Building Materials, Structures and Techniques, pp. 1067–1072. ISSN: 1877-7058. DOI: `https://doi.org/10.1016/j.proeng.2017.02.164`. URL: `https://www.sciencedirect.com/science/article/pii/S1877705817306707`.

[120] Duy-Nguyen Ta, Marin Kobilarov, and Frank Dellaert. “A factor graph approach to estimation and model predictive control on Unmanned Aerial Vehicles”. In: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2014, pp. 181–188. DOI: `10.1109/icuas.2014.6842254`.

[121] Amy Tabb and Khalil M. Ahmad Yousef. “Solving the Robot-World Hand-Eye(s) Calibration Problem with Iterative Methods”. In: *CoRR* abs/1907.12425 (2019). arXiv: `1907.12425`. URL: `http://arxiv.org/abs/1907.12425`.

[122] Wenfeng Tan et al. “A Comprehensive Calibration Method for a Star Tracker and Gyroscope Units Integrated System”. In: *Sensors* 18.9 (2018). ISSN: 1424-8220. DOI: `10.3390/s18093106`. URL: `https://www.mdpi.com/1424-8220/18/9/3106`.

[123] *The European Robotics Hackathon: ENRICH*. `https://enrich2021.european-robotics.eu/`. [Online; accessed 31st March 2022].

[124] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN: 0262201623 9780262201629. URL: `http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance\&n=283155\&n=507846\&s=books\&v=glance`.

[125] F. Tsushima et al. “Creation Of High Definition Map For Autonomous Driving”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* Xliii-b4-2020 (Aug. 2020), pp. 415–420. DOI: `10.5194/isprs-archives-XLIII-B4-2020-415-2020`.

[126] *Unity Real-Time Development Platform*. `https://unity.com/`. [Online; accessed 31st March 2022].

[127] *Unreal Engine: The most powerful real-time 3D creation tool*. `https://www.unrealengine.com`. [Online; accessed 31st March 2022].

[128] N. Vaskevicius et al. “Fast Detection of Polygons in 3D Point Clouds from Noise-Prone Range Sensors”. In: *IEEE International Workshop on Safety, Security and Rescue Robotics, SSRR*. Ieee. Rome, 2007, pp. 1–6.

[129] Jordi Vermeulen, Arne Hillebrand, and Roland Geraerts. "A comparative study of k -nearest neighbour techniques in crowd simulation". In: *Computer Animation and Virtual Worlds* 28 (May 2017), e1775. DOI: 10.1002/cav.1775.

[130] *VLP-16 User Manual*. https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf. [Online; accessed 31st March 2022].

[131] Li Wang, Zheng Zhang, and Ping Sun. "Quaternion-Based Kalman Filter for AHRS Using an Adaptive-Step Gradient Descent Algorithm". In: *International Journal of Advanced Robotic Systems* 12.9 (2015), p. 131. DOI: 10.5772/61313. eprint: https://doi.org/10.5772/61313. URL: https://doi.org/10.5772/61313.

[132] Steve T. Watt et al. "Understanding and applying precision time protocol". In: *2015 Saudi Arabia Smart Grid (SASG)*. 2015, pp. 1–7. DOI: 10.1109/sasg.2015.7449285.

[133] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[134] Zhengyou Zhang. "Camera Calibration". In: *Computer Vision: A Reference Guide*. Ed. by Katsushi Ikeuchi. Boston, MA: Springer US, 2014, pp. 76–77. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_164. URL: https://doi.org/10.1007/978-0-387-31439-6\%5F164.

[135] Zhengyou Zhang. "Parameter estimation techniques: a tutorial with application to conic fitting". In: *Image and Vision Computing* 15.1 (1997), pp. 59–76. ISSN: 0262-8856. DOI: https://doi.org/10.1016/S0262-8856(96)01112-2. URL: https://www.sciencedirect.com/science/article/pii/S0262885696011122.

[136] Lipu Zhou, Shengze Wang, and Michael Kaess. "DPLVO: Direct Point-Line Monocular Visual Odometry". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7113–7120. DOI: 10.1109/LRA.2021.3097052.

[137] Xingxing Zuo et al. "Robust Visual SLAM with Point and Line Features". In: *CoRR* abs/1711.08654 (2017). arXiv: 1711.08654. URL: http://arxiv.org/abs/1711.08654.

[138] Krzysztof Ćwian et al. "Large-Scale LiDAR SLAM with Factor Graph Optimization on High-Level Geometric Features". In: *Sensors* 21.10 (2021), p. 3445.