POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

CONDITIONALLY INDEPENDENT VISUAL SLAM WITH INTEGRATED BUNDLE ADJUSTMENT

Doctoral Dissertation of:
Simone Ceriani

Supervisor:
Prof. Matteo Matteucci

Tutor:
Prof. Andrea Bonarini

The Chair of the Doctoral Program:
Prof. Carlo Fiorini

2012 – XXV cycle

Abstract

The problem of Simultaneous Localization and Mapping (SLAM) regards the estimation of the pose of an observer (usually a robot) from sensor observations (i.e., local measurements), while creating, at the same time, a consistent map of the observed environment. Visual SLAM aims at the solution of the SLAM problem with the use of visual sensors, i.e., cameras, only.

In this thesis we worked on the development of a multicamera SLAM system able to operate in real time, on large environments with a generic number of cameras. The underlying methodology is the well known EKF-SLAM (Extended Kalman Filter SLAM) algorithm, but we introduced numerous improvements. First of all, we deeply reviewed the parameterizations for monocular SLAM presented in the literature, highlighting some properties that allow to reduce their computational complexity. Moreover, two new parameterizations have been introduced and their differences with previous parameterizations have been analyzed. A multicamera Visual SLAM system has been developed leveraging on monocular measurements, and allowing the definition of a flexible system in which cameras are treated independently, and, possibly, with reduced overlapping fields of view. The Conditionally Independent (CI) Submapping Framework is introduced in the system to treat large scale problem. Properties of this system are deeply investigated, bringing to light problems with ill conditioned matrices. To comply with these issues we reformulated the EKF SLAM algorithm in the Hybrid Indirect EKF SLAM form, were two common approaches to the Indirect EKF estimation (a.k.a. Error State EKF) are combined. As a final contribution, we developed a technique which allows to refine each submap of the CI-SLAM system with a Bundle Adjustement (BA) optimization. Optimized submaps are then reinserted in the CI-SLAM submaps collection, allowing the continuation of the estimation process. Extensive tests and analysis have been performed on simulated environments and on real datasets.

Contents

1 Introduction **1**
1.1 Summary . . . 1
1.2 State of Art and Related works . . . 3
1.3 Structure . . . 6

I Background Material **9**

2 3D Geometry **11**
2.1 Points, Vectors and Planes . . . 11
2.1.1 Points . . . 12
2.1.2 Vectors . . . 13
2.1.3 Relating Points and Vectors . . . 13
2.1.4 Vector products . . . 13
2.1.5 Points and Vectors in Homogeneous Coordinates . . . 14
2.1.6 Planes . . . 15
2.2 Reference Systems and Transformations . . . 15
2.2.1 Reference Systems . . . 16
2.2.2 Frame Transformations . . . 16
2.2.3 Points and Vectors Transformations . . . 17
2.2.4 Homogeneous Transformations . . . 17
2.2.5 Transformation Inversion . . . 18
2.2.6 Composition of Transformation . . . 18
2.3 Rotations Representation . . . 19
2.3.1 Euler Angles . . . 20
2.3.2 Axis/Angle . . . 21
2.3.3 Gibbs Vector and Modified Rodrigues Parameters . . . 21
2.3.4 Quaternions . . . 22

2.3.5 A Note on Small Rotations 24
2.3.6 Time Derivatives 24
2.3.7 Discussion on Rotation Representation 25

3 Computer Vision Introduction 27
3.1 Perspective Camera and Image Formation 27
3.1.1 Image Formation 28
3.1.2 Thin Lenses 29
3.1.3 The Pin-hole Model 31
3.1.4 Homogeneous Camera Matrix 32
3.1.5 Interpretation Line 34
3.1.6 Lens Distortion 35
3.1.7 Unified Mathematical Notation 36
3.1.8 Camera Calibration And Camera Reference System 37
3.2 Image Features 38
3.2.1 Feature detection 39
3.2.2 Feature Description and Matching 42

4 Filtering and Optimization 47
4.1 Brief Probability Recall 47
4.1.1 Axioms of Probability 47
4.1.2 Random Variables and Probability Density Function 48
4.1.3 Joint Probability 49
4.1.4 Random Vectors 50
4.1.5 Conditional Probability and Bayes Formula 51
4.1.6 Conditional Independence 52
4.1.7 Gaussian Random Variables and Vectors 52
4.1.8 Chi-Square Distributed Random Variables 53
4.2 Bayesian Filtering and the Extended Kalman Filter 54
4.2.1 General Bayes Filter 54
4.2.2 Kalman Filter 56
4.2.3 Extended Kalman Filter 58
4.3 Non Linear Least Squares Optimization 59
4.3.1 Non-Euclidean Spaces 61
4.3.2 Levenberg-Marquardt Algorithm 62

5 Simultaneous Localization And Mapping 63
5.1 SLAM Problem Statement 64
5.2 SLAM History and Applications 66
5.3 Probabilistic Formulation 67
5.4 EKF Based On-Line SLAM 69
5.4.1 State Vector 69
5.4.2 Initialization 69
5.4.3 Prediction Step 71
5.4.4 Landmark Addition 72
5.4.5 Measurement 73
5.4.6 Update Step 75

5.5 Graph Formulation . . . 77
5.5.1 Optimization On a Graph . . . 78
5.5.2 SLAM On a Graph . . . 80
5.5.3 Gauge of Freedom . . . 81
5.6 Conditionally Independent Sub-Maps SLAM . . . 82
5.6.1 From DBN to CI submap . . . 83
5.6.2 Map Transition Step . . . 88
5.6.3 Back-Propagation Step . . . 93
5.6.4 Loop closure . . . 96
5.6.5 Summary With An Example . . . 97

II Large Scale Real Time Visual SLAM 103

6 Mono and Multi Camera EKF Visual SLAM 105
6.1 Introduction . . . 105
6.2 Multi Camera EKF-SLAM With Conditional Independance Submapping . 107
6.2.1 The main loop . . . 107
6.2.2 Map management strategies . . . 110
6.3 Motion Models . . . 112
6.3.1 Motion Model With Odometric Information . . . 113
6.3.2 Motion Models Without Odometric Informations . . . 114
6.4 Parameterizations . . . 115
6.4.1 Definitions . . . 116
6.4.2 Inverse Scaling or Homogeneous Point - IS . . . 118
6.4.3 Unified Inverse Depth - UID . . . 123
6.4.4 Anchored Homogeneous Point - AHP . . . 128
6.4.5 Framed Homogeneous Point - FHP . . . 131
6.4.6 Framed Inverse Scale - FIS . . . 134
6.4.7 Remarks On The Parametrizations . . . 136

7 Indirect EKF SLAM 141
7.1 The Indirect or Error State EKF . . . 141
7.1.1 Hybrid EKF-SLAM . . . 143
7.2 Motion Models . . . 144
7.2.1 Motion Model With Odometric Informations . . . 146
7.2.2 Motion Model Without Odometric Informations . . . 149
7.3 Parameterization . . . 151
7.3.1 Anchor Frame . . . 152
7.3.2 Indirect Framed Homogeneous Point - FHP$_i$ . . . 155
7.3.3 Indirect Framed Inverse Scale - FIS$_i$ . . . 156

8 CIBA-SLAM: Conditional Independent Bundle Adjusted SLAM 159
8.1 From Gaussian Local Map To Graph Optimization . . . 160
8.1.1 DBN analysis . . . 161
8.1.2 Map Bundle Adjustment . . . 164
8.2 Graph Formalization . . . 165

8.2.1 Nodes in the graph . . . 166
8.2.2 The Odometric Edge . . . 168
8.2.3 Landmark Initialization Edge . . . 171
8.2.4 Landmark Measurement Edge . . . 171
8.2.5 Landmark Measurement In Multi Camera Setup . . . 172
8.2.6 Graph Initialization And Optimization Strategies . . . 172
8.2.7 Sub map setup . . . 174
8.3 From Graph Optimization To EKF . . . 177
8.3.1 Reconstruction Of The Gaussian Map . . . 179

9 Experimental Results **181**
9.1 Parameterization Evaluation . . . 181
9.1.1 Evaluation On Simulated Data . . . 182
9.1.2 Hybrid EKF SLAM Test with Loop Closure . . . 208
9.1.3 Evaluation On Real Data . . . 214
9.2 CIBA Slam Evaluation . . . 219
9.2.1 Simulated Scenario . . . 219
9.2.2 Annotated Dataset . . . 221
9.2.3 Real Experiments . . . 225
9.3 Time Performances Evaluation . . . 228
9.3.1 Simulated Experiments . . . 228
9.3.2 Real Experiments . . . 230
9.3.3 CIBA Slam . . . 231

10 Conclusions and Future Works **233**
10.1 CIBA SLAM System . . . 233
10.2 Future Works . . . 236

Bibliography **239**

III Appendices **247**

A Consistency Analysis **249**

CHAPTER 1

Introduction

1.1 Summary

The problem of Simultaneous Localization and Mapping (SLAM) regards the estimation of the pose of an observer (usually a robot) from sensor observations (i.e., local measurements), while creating, at the same time, a consistent map of the observed environment. Visual SLAM aims at the solution of the SLAM problem with the use of visual sensors, i.e., cameras, only; in particular, the problem takes the name of Monocular, Stereo and Trinocular SLAM when, respectively, one, two or three cameras are used, while the term multi-camera SLAM refers to a generic number of cameras. Visual SLAM has become a very attractive and active research field in the last decade due to the richness of the visual percepts coupled with the affordability and the low power consumption of the sensing device. In this thesis we work on the development of a multicamera SLAM system that is able to operate in real time, on large environments and with a generic number of cameras.

The first contribution of this thesis is the development of a generic and modular multi-camera SLAM system based on the well established EKF-SLAM (Extended Kalman Filter SLAM) approach, which uses an EKF as the estimator for the joint distribution of the observer pose and the environment map. Each camera is treated as an independent system that provides measurements, thus our system operates without any explicit reference to particular geometric constraints among cameras (e.g., the epipolar geometry for the stereo cameras or the trifocal tensor for trinocular vision). This approach guarantees a high level of modularity and flexibility: there are no building constraints on the overlapping of cameras fields of view and sensors can be added or removed at run time with the only requirement of synchronization and relative positioning. This system is inspired by the *BiCAM-SLAM* [91] approach proposed in the literature.

Monocular SLAM is the basic building block of the multi-camera SLAM system we develop and it is treated in the EKF SLAM framework through the use of *parameterizations* of landmarks. A second contribution of this thesis is the review of the different parametrizations which allow to treat the initial unknown depth of landmark points perceived by a single camera (a bearing only sensor) in the EKF SLAM framework. This review presents parameterizations from a novel perspective, highlighting the possibility of saving space in the state vector by sharing the common *anchor point* of landmarks initialized at the same time. This shrewdness, beside increasing the efficiency of the EKF-SLAM, avoids the introduction of singularities in the covariance matrix. During the review of parameterizations, we introduced also two novel parameterizations, named Framed Homogeneous Point (FHP) and Framed Inverse Scale (FIS), that extend the concept of anchor point to *anchor frame*. Anchor frames maintain in the filter state the past camera poses representing *milestones* along the trajectory. This gives us the base for the development of further contributions in the thesis. Anchor frame based parameterizations come at the cost of an increased computational complexity in case of FHP parametrization and an approximation on the estimation of the initial viewing ray in case of FIS parametrization.

The EKF-SLAM approach is not directly applicable when the size of the environment the robot (or the observer) is exploring becomes significantly large, since the computational requirements of the basic EKF algorithm increase with quadratic complexity preventing real time performance. To push our system toward real-time performance in very large environments, we rely on a solution already proposed in the literature, the *Conditional Independent Submaps (CI-SLAM)* framework, which allows linear complexity in large maps by decomposing the state into a set of conditional independent submaps. To the best of our knowledge, this framework has never been implemented before apart in the original work. Our implementation, although it does not cover all the features of the original framework, is generic and can be applied with any of the proposed parameterizations, resulting in a module that acts as a *plugin*, i.e., it is added to the multi camera SLAM system to extends its operability to large maps.

A third contribution comes from the introduction of the *Hybrid Indirect EKF-SLAM* approach; this approach models the SLAM problem in a *indirect form* by representing errors in estimates in the state vector instead of directly representing the values of interest. This approach allows to maintain a local representation of the variables involved in the estimation process, easing, in particular, the representation of rotations. The benefits of the indirect approach regard the state dimension reduction for the frame anchored parameterizations, where quaternions get substituted by rotation vectors, and the proper managing of covariance of rotations, avoiding singularities introduced by quaternions. The Hybrid EKF-SLAM is developed mixing the two classical approaches to the indirect EKF formulation, i.e., the *feedback* approach and the *feedforward* approach; the former is used for the state vector parts that represents the robot pose while the latter is used for the elements of the environment. Moreover, the Hybrid EKF-SLAM allows to apply the CI-SLAM framework procedures, while a pure feedback approach would have prevented the usage of this powerful submapping technique.

A final contribution is the development of a module that performs a refinement of the EKF state estimates through non linear optimization (i.e., Bundle Adjustment) applied on each submap of the CI-SLAM framework; this is possible leveraging on the novel parameterizations based on anchor frames. This module acts as a *plugin* of the whole multi camera SLAM system, thus it can be used or not, depending on the specific setup. The Bundle Adjustment is performed with a proper formulation of the optimization problem such that the results of the optimization may be reinserted in the EKF, allowing the continuation of the estimation and the propagation of non linear optimization improvement to the EKF machinery. We name the complete system CIBA-SLAM, Conditionally Independent Bundle Adjusted SLAM.

The complete system has been tested in simulation and on real data and it has proved to operate in real time on large environments. The main focus of this work was not on the implementation of a deployable system, but on the development of an EKF SLAM system that could be considered as a milestone towards a final implementation of a modular system able to operate in real time on a real robot, possibly on simple platforms. This motivated us to work on SLAM algorithms based on filtering techniques, although recent trends have moved the research toward pure optimization based strategies. We believe that filter based SLAM has still a role in this research field, especially in the direction of moving from fully fledged multi-core computers to low power embedded system, where high performance parallel processing, such as GPU (Graphics Processing Unit) based computation, can not be performed.

1.2 State of Art and Related works

The seminal solution to the SLAM problem is given in [87]. It uses an Extended Kalman Filter (EKF) as the central estimator for the map, represented by a vector of landmarks, and the current sensor pose. The estimate are represented as a multivariate Gaussian variable and the EKF estimates it through the prediction and the update steps. The prediction step aims at the movement of the sensor pose, the update step aims at the refinement of the sensor pose and landmarks estimates through measurements given by the sensor itself. The addition of an initialization step to the EKF machinery allows the environment exploration: new landmarks are added to the map when perceived for the first time by composing sensor measurements with the current sensor pose. The computation of the Jacobian of the addition function allows to compute the covariances of the new inserted elements with the rest of the map.

One of the first works implementing an on-line stereo SLAM working in real time is [21]. It used the Shi-Tomasi salient point detector and small patches as feature descriptors matched by correlation. One of the key aspect presented in this work is the *active search* approach to measurements: the EKF-SLAM mechanism allows to predict the projection of landmarks in the current image, thanks to the measurement step. By computing the covariance associated with the measure we can obtain a 2D elliptical region, in terms of Mahalanobis distance around the mean prediction, which bounds the search area in the image that contains the feature with some given probability. This approach reduce the research area, cutting down the computational cost and, to some extent, data association errors.

The system presented in [20] was the first on-line SLAM system working with a single camera, i.e., the first monocular SLAM system. The monocular approach is much more challenging than the stereo: a monocular camera is a bearing only sensor, i.e., it can perceive only the direction of the viewing ray of a projected environmental point, but it can not provide the distance of the observed point. The solution proposed in [20] uses a *delayed approach*, i.e., the landmarks are added as 3D points to the filter once a proper estimation of their depth, performed by an independent particle filter, has been reached. Consequently, the system needs to be started with a visible known pattern, in order to track the first movement up to the estimation of the depth of some new landmarks. This work can be considered the seminal paper in the monocular SLAM research field and it gave birth to an important research effort. Although it was the first on-line monocular SLAM system , most of its concepts and techniques are still a valid approach and the multicamera SLAM system presented in this work is highly based on the original proposal.

The first attempt to abandon the delayed initialization, i.e., to introduce landmarks in the filter state and use them in the measurement process from the first time they have been seen, was developed in [89] and [92], with the *Federate Information Sharing* approach. This system is an approximation of the Gaussian Sum Filter which introduces landmarks as a sum of Gaussian hypothesis along the viewing ray of the landmark. When landmarks are measured, the hypothesis are weighted by their likelihood and the less probable are pruned after a few steps.

The delayed initialization was definitely abandoned with the introduction of the Unified Inverse Depth Parameterization (UID), proposed in [65]. This work introduces the concept of *parameterization*, i.e., the description of a 3D point representing a landmark with a different formulation with respect to its Euclidean coordinates that allows to take in account the initial uncertainty on the unknown depth. The UID parameterization was successfully applied in numerous works, like [75], [67] and [100]. Since then, other parameterizations have been introduced, like the Inverse Scaling (IS) [62] [63] and the Anchored Homogeneous Point (AHP) [90] [93]. A good review of these parameterizations, among with an experimental evaluation of their properties, can be found in [90] and [93]. In this thesis we present two novel parameterizations, the Framed Homogeneous Point (FHP), originally presented in [10], and the Framed Inverse Scale (FIS). These parameterizations are not completely new: a proposal similar to FHP can be found in [43], while a proposal similar to FIS can be found in [74]. All the parameterizations are presented in this thesis in a different form with respect to their original formulation. Such a reformulation allows to point out a state vector size saving, thus a computational complexity reduction, and it solves some issues related to the use of the CI-SLAM framework with parameterizations.

In [91] an interesting discussion about the monocular approach to multi-camera systems was proposed with the provocative title *"BiCamSLAM: Two times mono is more than stereo"*. In conventional approaches, stereo cameras and multi camera systems are used taking into account the underlying mathematical and geometrical concepts (e.g., the epipolar geometry for stereo cameras and the trifocal tensor for trinocular cameras) to initialize 3D landmarks by triangulation of corresponding features. In the *BiCamSlam* approach the landmarks are added to the filter as if they are perceived by a monocular camera using the *Federate Information Sharing* approach. Subsequently measurements of landmarks performed in different cameras are used to estimate the depth of the landmarks in an implicit way: each observation corresponds to a measurement equation of the system. This gave to the system an intrinsic flexibility: a landmark can be added to the filter also if it has been perceived by a single camera, while in a standard stereo approach it is needed to perceive it in both images to perform triangulation. Moreover, the landmark can be retrieved independently in each camera of the system. This implies that the estimate are updated even when the landmark is perceived in a single image and this is is particularly useful when field of view of cameras are only partially overlapped. In a conventional stereo approach landmarks needs to be matched in both cameras to represent a measure for the system. This thesis extends the BiCamSlam approach to a multi camera system leveraging on the use of parameterizations, which were not already introduced when this approach was originally proposed. Thanks to this we develop a system that rapidly estimates depth of points when double, in a stereo setup, or multiple, in a multi camera setup, measurements of landmarks are available and it is able to deal with bearing only (i.e., monocular) measurements too. Such a system results very flexible due to its modularity: cameras can be just added or dropped from the system when they are needed. A possible application of this system is an active SLAM algorithm that turn on or off cameras when needed, augmenting the precision of the estimation or saving computational power.

The EKF-SLAM algorithm suffers from some limitations. First, the complexity of the algorithm is dominated by the update step, which is $O(n^2)$ where n is the number of landmarks in the map. This implies that when the number of landmarks grows, i.e., the environment to explore is not limited to an experimental small setup, EKF-SLAM looses quickly the ability to run in real time. Secondly, EKF formulation uses a linear approximations for the Jacobians and this produce optimistic estimation for the covariance matrix that usually results in an inconsistent behavior of the filter. *Sub-Mapping* techniques comes the overcome both limitations by splitting the explored area into several subproblems, each of them with a bounded number of landmarks. This implies that the computational complexity is bounded and can be considered constant for each submap. The *Conditional Independence Sub-Maps* (CI) framework, presented in [78] and [77], represents one of the smartest and most effective sub-mapping technique. This framework allows to split the entire map in several small local submaps which share landmarks that can be observed by more than one map. The term *local* indicates that each submap of the problem store landmarks in a local reference frame, while the transformation that links the local reference frame with the global one is stored apart. Besides reducing the computational complexity, the usage of small local submaps bound the uncertainty, being each submap started in the origin with zero uncertainty and this reduces the approximation introduced by linearizations. The convenience of the CI-SLAM technique lies in the ability to perform large scale SLAM, i.e., to operate on real dataset, without introducing any approximations besides the inherent EKF linearizations. Thanks to this properties, the CI-SLAM framework results the best submapping framework for submaps represented as Gaussian multivariate variable, while other approaches, such that [6] and [17], introduces approximation when they split the problem in submaps.

Filtering approaches model the problem as an on-line state estimation where the state of the system consists in the current robot position and the map. Conversely, *smoothing* approaches estimate the full trajectory of the robot from the full set of measurements. These approaches address the so-called *full SLAM* problem and they typically rely on least-square error minimization techniques. An intuitive way to address the full SLAM problem is via its so-called graph-based formulation. Solving a graph-based SLAM problem involves to construct a graph whose nodes represent robot poses and landmarks and in which an edge between two nodes encodes a sensor measurement that constrains the connected poses. The solution of the SLAM problem corresponds to find a configuration of the nodes that is maximally consistent with the measurements. This involves solving a large error minimization problem. The graph-based formulation of the SLAM problem dates back in 1997 [55]. However, it took several years to make this formulation popular due to the high complexity of solving the error minimization problem using standard techniques. Recent insights into the structure of the SLAM problem and advancements in the fields of sparse linear algebra resulted in efficient approaches to the optimization problem. Consequently, graph-based SLAM methods have undergone a renaissance and currently belong to the state-of-the-art techniques [49], [35]. These systems have been successfully applied to the visual SLAM problem and we can cite the impressive real time application for augmented reality in small environments presented in [46] and [47] named *Parallel tracking and mapping (PTAM)*. However, applications that use optimization techniques (a.k.a. Bundle Adjustment techniques) for Visual SLAM are still at an early stage and their scalability to large maps is still under investigation, although some interesting results on real datasets are available [95] [97]. Recent comparison between filter based techniques and optimization based strategies seem to tip the scale in favor of the latter, as supported by [96] and [98].

In our opinion there are still some reason to push forward research in the filter based Visual SLAM, especially if we take into account standard architecture and low power CPU. In this work we choose to rely on consolidate system, aiming at the development of a modular Visual SLAM system that can be configured to different setups (e.g., monocular, stereo or multi-camera) and with different target platform (e.g., on real robots with low power computers or standard PC architectures). Consequently we choose to introduce a Bundle Adjustment module that works only at a local level, i.e., it refines the estimate of each submap of the CI-SLAM system. The global map, subdivided in submaps, is still maintained by the CI-SLAM algorithm. Since the optimization algorithm is integrated with the CI-SLAM system, it can be seen as a *plugin* of the entire system: if the system has a sufficient computational power it can be used to refine the solutions, otherwise it can be just turned off. We call the entire system developed in this thesis “Conditionally Independent Bundle Adjusted SLAM (CIBA SLAM)”.

1.3 Structure

The remaining of this thesis is divided in three parts: Part I (Chapters 2 to 5) represents the basis for the development of the thesis and for the comprehension of a complete Visual SLAM system, Part II (Chapters 6 to Chapter 10) defines the components of the Visual SLAM system developed during this work and presents experimental results, conclusion and future works; Part III (Appendix A) contains tables with detailed results of consistency analysis performed in Chapter 9. In details,

Chapter 2 introduces 3D geometry, using the notation that is used in the remaining of the thesis. In particular 3D points and vector and their relative operations are recalled; frame and reference system relations and transformations are defined and different formulation for rotation representation are introduced.

Chapter 3 gives a short introduction to computer vision techniques defining the perspective camera and its mathematical model; it introduces some basic algorithms for image analysis, referring in particular to feature extraction and tracking.

Chapter 4 introduces the reader to probability theory by recalling basic concepts with a special attention to Gaussian variables and their properties. Bayesian Filtering in the realm of the Gaussian filtering is explained by the description of the Kalman Filter (KF) and its Extended version (EKF). Lastly, an introduction to Non Linear Least Square (NNLS) optimization techniques and their application to non Euclidean spaces is provided.

Chapter 5 introduces the SLAM (Simultaneous Localization And Mapping) problem. Starting from the probabilistic formulation of the SLAM problem, the EKF-SLAM algorithm is described in a generic way, detailing the key steps, such as landmark addition and measurement procedures, and describing the arrangements that simplify computation taking advantage from sparsity in the matrices. An alternative formulation of the SLAM problem, represented as a graph, is then presented and the strategies that can be adopted to optimize the graph using the NNLS techniques (namely, the Bundle Adjustment algorithms) are presented. Lastly, the Conditionally Independent Sub-Mapping SLAM framework, a technique that allows to handle large problems with the EKF SLAM algorithm, is presented.

Chapter 6 describes a first contribution of the thesis: a Multicamera SLAM system based on the monocular EKF SLAM algorithm. Different motion model that can be used in the prediction step of the EKF are presented together with the parameterizations introduced in the literature for the monocular EKF SLAM algorithm. Moreover, some properties of parametrization are highlighted and two new parameterizations, named Framed Homogeneous Point (FHP) and Framed Inverse Scale (FIS) are presented.

Chapter 7 introduces a different way of thinking the EKF estimation mechanism: the so called Indirect EKF or Error State EKF. This approach has allowed us to introduce the Hybrid Indirect EKF SLAM, a variant of the Indirect EKF that, thanks to the reformulation of the FHP and FIS parameterizations, allows to solve some issues related to rotation representation in EKF SLAM.

Chapter 8 describes the last contribution of this thesis: a module that performs Bundle Adjustment on the estimation performed by the Hybrid Indirect EKF SLAM system. This module is integrated with the CI SLAM framework: it takes initializations from the estimates performed by the EKF SLAM system, optimizes them with Bundle Adjustment techniques and propagates the result to the EKF SLAM system, resulting transparent to the CI SLAM system.

Chapter 9 is aimed at the evaluation of the proposed system. Simulated experiments are used to evaluate the different parameterizations and their impact on the consistency; real experiments propose an evaluation of the proposed system and its time performances on datasets collected by real robots.

Chapter 10 draws the conclusion of this works, resuming all the contribution provided by this thesis and it traces some possible future works and research directions.

Part I

Background Material

CHAPTER 2

3D Geometry

In this chapter basic notions about 3D geometry, transformations and time differentiation are introduced in order to provide a good basis for the comprehension of next chapters and to fix a common notation for the rest of the thesis. In particular, points and vector are introduced firstly in Euclidean coordinates, then in Homogeneous coordinates; then 3D planes are described in Homogeneous coordinates, which results very convenient for this. Reference systems and frame transformations are described in terms of rototranslations expressed in Homogeneous coordinates; the rules for transformation composition and inversion are finally explained. Particular attention is given in the description of rotations, introducing the Euler Angles formalism, the Axis/Angle notation and the three formulations derived from it: the Rodrigues parameters, the Gibbs vector and the Modified Rodrigues parameters. Moreover, the use of unit Quaternions for rotation representation is described. The case of small rotations is treated by the introduction of Taylor first order expansion of rotation compositions both for rotation matrices and quaternion representations.

2.1 Points, Vectors and Planes

Points are the simplest geometric entity and they represent locations in the space. Vectors, although they are often confused with points, code directions and distances.

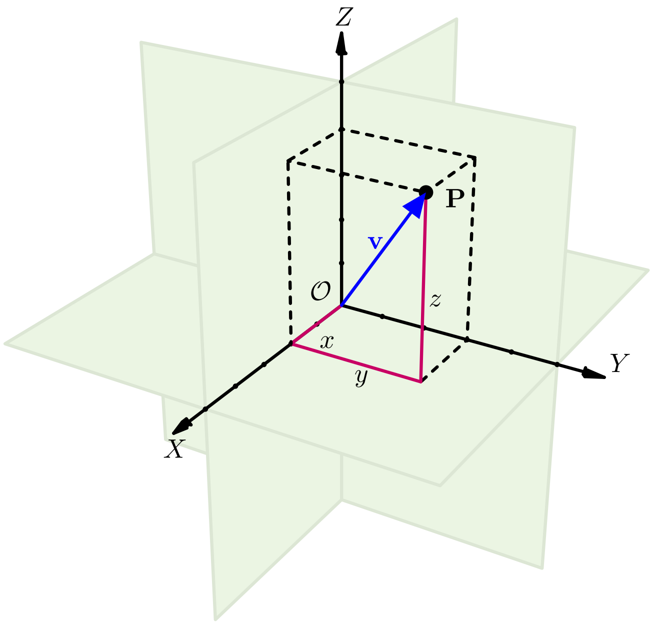


Figure 2.1: *A point* $\mathbf{P}$ *in Cartesian space with its coordinates measured with respect to the origin* $\mathcal{O}$ *and the vector* $\mathbf{v} = \mathbf{P} - \mathbf{O}$ *applied to the origin itself.*

2.1.1 Points

The three dimensional Euclidean space is a geometric model of the physical space around us. A fourth dimension that is not represented in this model is time. The three dimensions are commonly referred as *length*, *width* and *height* (or *depth*) when they are used to describe object dimensions. When the three dimensions are used to represent a location in space, a common convention is to describe the distance from the *origin* ($\mathcal{O}$) of a *Cartesian coordinate system*. The origin, represented by the calligraphic letter $\mathcal{O}$, is located at the point of intersection of the *Cartesian axes* ($\mathbf{O} = [0, 0, 0]^T$). With this convention we refers the axis direction too, i.e., a complete Cartesian reference system. Cartesian axes (X, Y, Z) are ordered triplets of lines, any two of them being perpendicular; a single unit length is uniquely defined for all the three axes. The location of a *point* $\mathbf{P}$ is represented by a triplet $[x, y, z]^T \in \mathbb{R}^3$. The components of the triplet are usually referred as x, y and z and each of them represents the signed distance from the three planes defined by the remaining couple of axes (e.g., the x coordinate is the distance of $\mathbf{P}$ from the $y - z$ plane). Figure 2.1 shows an example of a point in Cartesian coordinates and its distance from the three planes. Points represent locations in the space and they are usually drawn as *dots*. When the Cartesian reference system is not obvious, point notation is enhanced referring to the origin as $\mathbf{P}^{\mathcal{O}}$.

2.1.2 Vectors

Vectors in Cartesian space are often confused with points, but they are very different: while points are, namely, locations in space, a vector represents a *direction* and a *distance* (also called *magnitude* or *module*). A vector $\mathbf{v}$ in 3D is still represented by three elements ($\mathbf{v} = [\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z]^T$), but it is by definition *free*, i.e., it could be applied to any location in space. Generally, vectors are drawn by *arrows* starting at a fixed location in space, i.e., they are applied to a reference point. When the Cartesian reference system is not obvious, vector notation is enhanced with the chosen origin as superscript: $\mathbf{v}^{(\mathcal{O})}$. The magnitude of a 3D vector is $\|\mathbf{v}\| = \sqrt{\mathbf{v}_x^2 + \mathbf{v}_y^2 + \mathbf{v}_z^2}$ and a $\overline{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ is the *unit vector*. A unit vector has magnitude one and it codes a direction.

2.1.3 Relating Points and Vectors

The relationships between points and vectors are described by the following axioms:

- For each pair of points $\mathbf{P}$, $\mathbf{Q}$ there exists an unique vector $\mathbf{v}$ such that

  $$\mathbf{v} = \mathbf{P} - \mathbf{Q}. \tag{2.1}$$

  i.e., vector $\mathbf{v}$ can be generated by the difference of two locations in the space (points).

- For each point $\mathbf{Q}$ and vector $\mathbf{v}$ there exists an unique point $\mathbf{P}$ such that

  $$\mathbf{P} = \mathbf{Q} + \mathbf{v}; \tag{2.2}$$

  i.e., the point $\mathbf{P}$ could be reached by moving from point $\mathbf{Q}$ a distance $\|\mathbf{v}\|$ in direction $\overline{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$.

- Given three points $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$, these points satisfy

  $$\mathbf{P} - \mathbf{R} = (\mathbf{P} - \mathbf{Q}) + (\mathbf{Q} - \mathbf{R})\,; \tag{2.3}$$

  i.e., naming $\mathbf{v_{RP}} = \mathbf{P} - \mathbf{R}$, $\mathbf{v_{QP}} = \mathbf{P} - \mathbf{Q}$ and $\mathbf{v_{RQ}} = \mathbf{Q} - \mathbf{R}$, it is possible to write the rule for vector composition:

  $$\mathbf{v_{RP}} = \mathbf{v_{QP}} + \mathbf{v_{RQ}} = \mathbf{v_{RQ}} + \mathbf{v_{QP}}. \tag{2.4}$$

2.1.4 Vector products

There are two type of products for vectors: the *scalar product* or *inner product* and the *cross product* or *outer product*. The *scalar product* of $\mathbf{v}$ and $\mathbf{u}$ is explicitly indicated with the $\cdot$ symbol and could be performed by standard matrix product:

$$\mathbf{v} \cdot \mathbf{u} = \mathbf{v}^T\mathbf{u}. \tag{2.5}$$

It could be also expressed as

$$\mathbf{v} \cdot \mathbf{u} = \|\mathbf{v}\|\|\mathbf{u}\| \cos(\theta), \tag{2.6}$$

where θ is the angle between the two vectors. This implies that $\mathbf{v} \cdot \mathbf{u} = 0$ if $\mathbf{v}$ is perpendicular to $\mathbf{u}$. Notice that $\|\mathbf{v}\| = \sqrt{\mathbf{v} \cdot \mathbf{v}}$.

The *cross product* of two vectors $\mathbf{v}$ and $\mathbf{u}$ results in a vector $\mathbf{w}$ which is perpendicular to both vectors, being perpendicular to the plane containing them. It is indicated by $\times$ operator and results in

$$\mathbf{w} = \mathbf{v} \times \mathbf{u} = \|\mathbf{v}\|\|\mathbf{w}\| \sin(\theta)\overline{\mathbf{n}}, \tag{2.7}$$

where θ is the angle between the two vectors and $\overline{\mathbf{n}}$ is the unit vector normal to the plane containing $\mathbf{v}$ and $\mathbf{u}$ in the direction given by the *right-hand rule* that will be introduced in Section 2.2.1. Analytically, the cross product could be expressed by

$$\mathbf{v} \times \mathbf{u} = \det \begin{bmatrix} \bar{\mathbf{i}} & \bar{\mathbf{j}} & \bar{\mathbf{k}} \\ \mathbf{v}_x & \mathbf{v}_y & \mathbf{v}_z \\ \mathbf{u}_x & \mathbf{u}_y & \mathbf{u}_z \end{bmatrix} = \begin{bmatrix} \mathbf{v}_y\mathbf{u}_z - \mathbf{u}_y\mathbf{v}_z \\ \mathbf{v}_z\mathbf{u}_x - \mathbf{u}_z\mathbf{v}_x \\ \mathbf{v}_x\mathbf{u}_y - \mathbf{u}_x\mathbf{v}_y \end{bmatrix}, \tag{2.8}$$

where $\bar{\mathbf{i}}$, $\bar{\mathbf{j}}$ and $\bar{\mathbf{k}}$ are respectively the unit vector that describe the directions of X, Y and Z axes. Alternatively, the cross product can be expressed by introducing the operator $[\cdot]_\times$:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -\mathbf{v}_z & \mathbf{v}_y \\ \mathbf{v}_z & 0 & -\mathbf{v}_x \\ -\mathbf{v}_y & \mathbf{v}_x & 0 \end{bmatrix}, \tag{2.9}$$

it can be defined as

$$\mathbf{v} \times \mathbf{u} = [\mathbf{v}]_\times \mathbf{u}. \tag{2.10}$$

Cross product is not commutative, in particular $\mathbf{v} \times \mathbf{u} = -\mathbf{u} \times \mathbf{v}$ and, in matrix form, $[\mathbf{v}]_\times \mathbf{u} = [\mathbf{u}]_\times^T \mathbf{v} = -[\mathbf{u}]_\times \mathbf{v}$. Useful properties of $[\cdot]_\times$ are related to its powers:

$$[\mathbf{v}]_\times^2 = \|\mathbf{v}\|^2 \left(\overline{\mathbf{v}}\,\overline{\mathbf{v}}^T - \mathbf{I}\right), \tag{2.11}$$

$$[\mathbf{v}]_\times^3 = -\|\mathbf{v}\|^3 \left[\overline{\mathbf{v}}\right]_\times. \tag{2.12}$$

2.1.5 Points and Vectors in Homogeneous Coordinates

In the Cartesian coordinate system infinity could not be explicitly represented. Homogeneous coordinates allow to represent points at infinity using finite coordinates. This is achieved by adding a coordinate to the number of dimensions of the space which has to be represented. A 3D point in homogeneous coordinates will be represented by

$$\mathbf{P}_h = [x, y, z, \omega]^T, \tag{2.13}$$

and it is equivalent to the Cartesian point

$$\mathbf{P} = \left[\frac{x}{\omega}, \frac{y}{\omega}, \frac{z}{\omega}\right]^T, w \neq 0. \tag{2.14}$$

The introduction of an additional degree of freedom in the representation makes homogeneous points invariant to scale transformations: considering a point $\mathbf{P}_h = [x, y, z, \omega]^T$ and a scalar element $\lambda \neq 0$, the 3D points represented by $\mathbf{P}_h$ and $\lambda\mathbf{P}_h$ are equivalent,

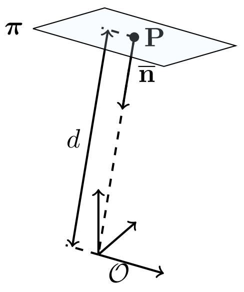


Figure 2.2: *Description of a plane $\boldsymbol{\pi}$ through its normal unit vector $\overline{\mathbf{n}}$ and its distance from the origin d.*

i.e., a single point can be represented by infinitely many homogeneous coordinates. Moreover, points at the infinity (or improper points) could be represented with $\omega = 0$: intuitively, when ω approach to zero, the point $[x, y, z, \omega]^T$ moves farther away in the direction $[x, y, z]$, reaching the infinity when $\omega = 0$. Points at the infinity allow to easily code *directions* and they express lines in parametric form: the line starting from a finite point $\mathbf{P} = [\mathbf{P}_x, \mathbf{P}_y, \mathbf{P}_z, \mathbf{P}_\omega]$ in the direction $\mathbf{D} = [\mathbf{D}_x, \mathbf{D}_y, \mathbf{D}_z, 0]$ is given by $\mathbf{P} + \alpha\mathbf{D}$. When $||\mathbf{D}|| = 1$ and $\mathbf{P}_\omega = 1$, α represents the Euclidean distance from $\mathbf{P}$.

Notice that the point $[0, 0, 0, 0]^T$ has no meaning, thus 3D homogeneous coordinates are defined in $\mathbb{R}^4 - [0, 0, 0, 0]^T$. Similar reasoning could be applied to a 2D Cartesian space: it is represented in $\mathbb{R}^3 - [0, 0, 0]^T$ homogeneous space. Italic bold letters (e.g., $\boldsymbol{p} = [x, y] \equiv \lambda[x, y, 1]$ are used to identify points in 2D Cartesian space or in 2D homogeneous coordinates.

2.1.6 Planes

Planes are easily described in Homogeneous coordinates with a 3D homogeneous coordinate element, i.e., with a 4 element vector $\boldsymbol{\pi} = [a, b, c, d]^T$. It is worth to notice that the description of planes and points in Homogeneous coordinates share the same notations, i.e., a 4 element vector describes both of them, but the interpretation of the elements, obviously, differs. Planes inherit the properties of Homogeneous coordinates, thus there are infinitely many equivalent representation of the same plane constructed by scaling it: $\boldsymbol{\pi} \equiv \lambda\boldsymbol{\pi}$ with $\lambda \neq 0$. Considering a 3D point in Homogeneous coordinates $\mathbf{P} = [x, y, z, w]^T$, it lies on the plane $\boldsymbol{\pi}$ if and only if $\boldsymbol{\pi}^T\mathbf{P} = \mathbf{P}^T\boldsymbol{\pi} = 0$. All the improper points (i.e., points at the infinity with $w = 0$) lie on the plane $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$, which is the plane at the infinity. As for Homogeneous points, the planes are defined on $\mathbb{R}^4 - [0, 0, 0, 0]^T$. Without loss of generality, a plane can be easily coded by a unit vector $\overline{\mathbf{n}}$ normal to it and by its distance d from the reference system origin as $\boldsymbol{\pi} = [\overline{\mathbf{n}}^T d]^T$, as shown in Figure 2.2.

2.2 Reference Systems and Transformations

More than one Cartesian reference system could be defined in the Euclidean space. Consequently a point location (or any other geometric entity) is defined with respect to a specific reference system (or *origin* or *frame*); relations between different origins (i.e., relative position and axes orientation) allow points to be expressed in a different reference system than the original.

2.2.1 Reference Systems

Origins (or frames or reference systems) will be indicated with a calligraphic symbol, e.g. $\mathcal{O}$, $\mathcal{W}$ or $\mathcal{R}$. For the purpose of this work we will use the *right-handed* Cartesian reference system: considering the index finger of the right hand is pointed forward, the middle finger bent inward at a right angle to it, and the thumb placed at a right angle to both, the three fingers indicate the relative directions of the X, Y, and Z-axes. Conversely, if the same is done with the left hand, a *left-handed system* results.

The right-handed reference system convention establishes the direction of positive rotations. Let consider a rotation by angle θ around an axis. θ is positive if, putting the Z axis of the right-handed reference system in coincidence with the rotation axis, the X axis moves in the direction of Y. This rule is know with the name of *right-hand thumb* or *corkscrew* rule.

2.2.2 Frame Transformations

Let consider two different Cartesian reference systems $\mathcal{O}$ and $\mathcal{W}$ and let name $\mathbf{O}^{(\mathcal{O})}$ and $\mathbf{W}^{(\mathcal{W})}$ their origin points respectively in their reference system (see Figure 2.3). The relative pose of $\mathcal{O}$ with respect to $\mathcal{W}$ is indicated by $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$, that is composed by a translation $\mathbf{t}_{\mathcal{O}}^{\mathcal{W}}$ and a rotation $\mathbf{R}_{\mathcal{O}}^{\mathcal{W}}$. The existing transformation between frames is a *roto-translation*, also called *isometry*.

The simplest way to encode the translation is provided by considering $\mathbf{t}_{\mathcal{O}}^{\mathcal{W}}$ as a vector that encodes $\mathbf{O}^{(\mathcal{W})}$, i.e. the displacement of the origin $\mathcal{O}$ with respect to $\mathcal{W}$, equivalent to $\mathbf{O}^{(\mathcal{W})} = \mathbf{W}^{(\mathcal{W})} + \mathbf{t}_{\mathcal{O}}^{\mathcal{W}}$.

Rotations in 3D are naturally encoded by a *rotation matrix*, i.e., a 3×3 orthonormal matrix $\mathbf{R}$ with these properties:

- $\mathbf{R}^{-1} = \mathbf{R}^T$.
- $\det(\mathbf{R}) = 1$.
- $\mathbf{R} = \begin{bmatrix} \overline{\mathbf{u}} & \overline{\mathbf{v}} & \overline{\mathbf{w}} \end{bmatrix}$, i.e., each 3×1 column vector is a unit vector
- $\overline{\mathbf{u}} \cdot \overline{\mathbf{v}} = \overline{\mathbf{v}} \cdot \overline{\mathbf{w}} = \overline{\mathbf{u}} \cdot \overline{\mathbf{w}} = 0$, i.e., any two of the column vectors are perpendicular.
- $\overline{\mathbf{u}} \times \overline{\mathbf{v}} = \overline{\mathbf{w}}$, i.e., the three columns vectors form a right handed reference system.
- $\mathbf{R} = \begin{bmatrix} \overline{\mathbf{r}}^T \\ \overline{\mathbf{s}}^T \\ \overline{\mathbf{t}}^T \end{bmatrix}$, i.e., each row is a unit vector.

The more intuitive way of reading the $\mathbf{R}_{\mathcal{O}}^{\mathcal{W}}$ matrix is to take columns as the direction of the axis of frame $\mathcal{O}$ with respect to $\mathcal{W}$. Take as an illustrative example the Figure 2.3: the frame $\mathcal{O}$ is placed at point $[\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z]$ in $\mathcal{W}$ reference system and X axis of $\mathcal{O}$ is directed

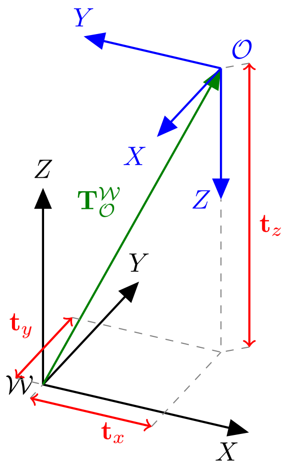


Figure 2.3: *Example of relative placement of frames. The pose of the origin of $\mathcal{O}$ w.r.t. $\mathcal{W}$ is represented by $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$ and it is composed by $\mathbf{t}_{\mathcal{O}}^{\mathcal{W}} = [\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z]$ (position of the point $\mathbf{O}^{(\mathcal{W})}$) and $\mathbf{R}_{\mathcal{O}}^{\mathcal{W}}$ (the directions of axes of $\mathcal{O}$ w.r.t. the axes of $\mathcal{W}$)*

as the $-Y$ of $\mathcal{W}$, Y axis of $\mathcal{O}$ is directed as the $-X$ of $\mathcal{W}$ and Z axis of $\mathcal{O}$ is directed as the $-Z$ of $\mathcal{W}$. Thus, $\mathbf{t}_{\mathcal{O}}^{\mathcal{W}} = [\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z]$ and $\mathbf{R}_{\mathcal{O}}^{\mathcal{W}} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

2.2.3 Points and Vectors Transformations

Let consider the same scenario of the previous section with two origin $\mathcal{W}$ and $\mathcal{O}$ related by $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$. Suppose to have a point $\mathbf{P}^{(\mathcal{O})}$, i.e., a location relative to the reference system $\mathcal{O}$. The same point expressed in the $\mathcal{W}$ reference system is

$$\mathbf{P}^{(\mathcal{W})} = \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} \mathbf{P}^{(\mathcal{O})} + \mathbf{t}_{\mathcal{O}}^{\mathcal{W}}. \tag{2.15}$$

This implies that $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$ could be read as *the transformation which transforms a point from the $\mathcal{O}$ reference system to the $\mathcal{W}$ reference system.*

Vectors code directions and distances and they are *free*, i.e., they could be applied to any location in space. This implies that, to express a vector $\mathbf{v}^{(\mathcal{O})}$ in a different reference frame $\mathcal{W}$, only the rotation operation has to be applied:

$$\mathbf{v}^{(\mathcal{W})} = \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} \mathbf{v}^{(\mathcal{O})}. \tag{2.16}$$

2.2.4 Homogeneous Transformations

Homogeneous coordinates simplify the transformations introduced in the previous sections. A frame transformation $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$, composed by $\mathbf{R}_{\mathcal{O}}^{\mathcal{W}}$ and $\mathbf{t}_{\mathcal{O}}^{\mathcal{W}}$ is expressed, in Homogeneous coordinates, by a Homogeneous 4×4 matrix

$$\begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} & \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix} . \tag{2.17}$$

The transformation of an Homogeneous point $\mathbf{P}_h^{(\mathcal{O})} = [\mathbf{P}_x^{(\mathcal{O})}, \mathbf{P}_y^{(\mathcal{O})}, \mathbf{P}_z^{(\mathcal{O})}, \mathbf{P}_\omega^{(\mathcal{O})}]^T = [\mathbf{P}_{\mathbf{xyz}}^{(\mathcal{O})^T}, \mathbf{P}_\omega^{(\mathcal{O})}]^T$ results in

$$\mathbf{P}_h^{(\mathcal{W})} = \begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} & \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\mathbf{xyz}}^{(\mathcal{O})} \\ \mathbf{P}_\omega^{(\mathcal{O})} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} \mathbf{P}_{\mathbf{xyz}}^{(\mathcal{O})} + \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \mathbf{P}_\omega^{(\mathcal{O})} \\ \mathbf{P}_\omega \end{bmatrix} . \tag{2.18}$$

When $\mathbf{P}^{(\mathcal{O})}$ is a real point (i.e., $\mathbf{P}_\omega^{(\mathcal{O})} \neq 0$) Equation 2.18 corresponds to the transformation of the Cartesian point as from Equation 2.15, while when $\mathbf{P}^{(\mathcal{O})}$ is an improper point (i.e., it encodes a direction with $\mathbf{P}_\omega^{(\mathcal{O})} = 0$) it corresponds to the vector transformation as from Equation 2.16. Notice that transformation of points by the mean of isometries maintains improper points at infinity and finite points in finite range.

2.2.5 Transformation Inversion

Given $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$, the inverse transformation $\mathbf{T}_{\mathcal{W}}^{\mathcal{O}} = \mathbf{T}_{\mathcal{O}}^{\mathcal{W}^{-1}}$, expressed in the two components $\mathbf{t}_{\mathcal{W}}^{\mathcal{O}}$ and $\mathbf{R}_{\mathcal{W}}^{\mathcal{O}}$, is

$$\mathbf{R}_{\mathcal{W}}^{\mathcal{O}} = \mathbf{R}_{\mathcal{O}}^{\mathcal{W}^{-1}} = \mathbf{R}_{\mathcal{O}}^{\mathcal{W}^T}, \tag{2.19}$$

$$\mathbf{t}_{\mathcal{W}}^{\mathcal{O}} = -\mathbf{R}_{\mathcal{O}}^{\mathcal{W}^T} \mathbf{t}_{\mathcal{O}}^{\mathcal{W}}. \tag{2.20}$$

In homogeneous coordinates this results in the following 4×4 matrix

$$\begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}^T} & -\mathbf{R}_{\mathcal{O}}^{\mathcal{W}^T} \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix} , \tag{2.21}$$

that is the inverse of the homogeneous matrix which represents $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$.

Given $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$ and a Homogeneous point $\mathbf{P}^{(\mathcal{W})}$, it is possible to express $\mathbf{P}^{(\mathcal{O})}$ as

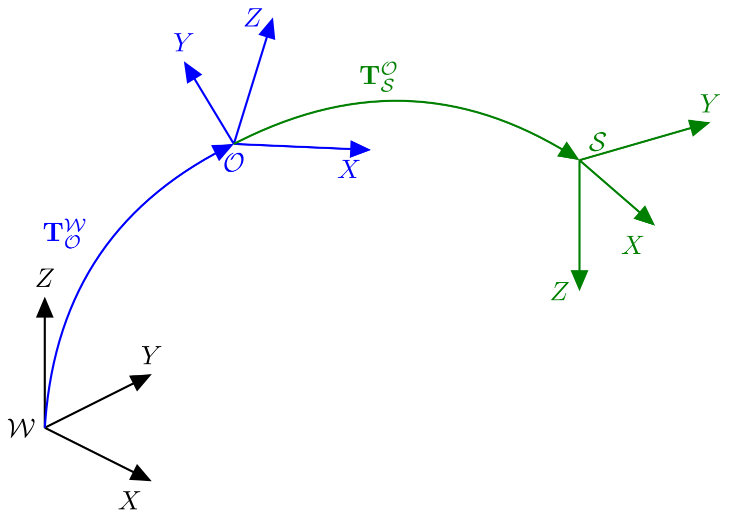
$$\mathbf{P}^{(\mathcal{O})} = \mathbf{T}_{\mathcal{W}}^{\mathcal{O}} \mathbf{P}^{(\mathcal{W})} \tag{2.22}$$

$$= \mathbf{T}_{\mathcal{O}}^{\mathcal{W}^{-1}} \mathbf{P}^{(\mathcal{W})} \tag{2.23}$$

$$= \begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}^T} \left(\mathbf{P}_{\mathbf{xyz}}^{(\mathcal{W})} - \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \mathbf{P}_\omega^{(\mathcal{W})} \right) \\ \mathbf{P}_\omega \end{bmatrix} . \tag{2.24}$$

2.2.6 Composition of Transformation

Without loss of generality, let us consider the case of Figure 2.4 with three origins in the Euclidean space: $\mathcal{W}$, $\mathcal{O}$ and $\mathcal{S}$. The known relations are $\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$ and $\mathbf{T}_{\mathcal{S}}^{\mathcal{O}}$. The transformation



Y
Z
$\mathbf{T}_{\mathcal{S}}^{\mathcal{O}}$
Y
$\mathcal{S}$
$\mathcal{O}$
X
X
Z
$\mathbf{T}_{\mathcal{O}}^{\mathcal{W}}$
Z
Y
$\mathcal{W}$
X

Figure 2.4: *Example of transformation composition for three frames $\mathcal{W}$, $\mathcal{O}$ and $\mathcal{S}$.*

$\mathbf{T}_{\mathcal{S}}^{\mathcal{W}}$, considering homogeneous matrices, is then given by:

$$\mathbf{T}_{\mathcal{S}}^{\mathcal{W}} = \mathbf{T}_{\mathcal{O}}^{\mathcal{W}}\mathbf{T}_{\mathcal{S}}^{\mathcal{O}} \tag{2.25}$$

$$\begin{bmatrix} \mathbf{R}_{\mathcal{S}}^{\mathcal{W}} & \mathbf{t}_{\mathcal{S}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}} & \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{S}}^{\mathcal{O}} & \mathbf{t}_{\mathcal{S}}^{\mathcal{O}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.26}$$

$$= \begin{bmatrix} \mathbf{R}_{\mathcal{O}}^{\mathcal{W}}\mathbf{R}_{\mathcal{S}}^{\mathcal{O}} & \mathbf{R}_{\mathcal{O}}^{\mathcal{W}}\mathbf{t}_{\mathcal{S}}^{\mathcal{O}} + \mathbf{t}_{\mathcal{O}}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.27}$$

Thanks to this composition rule and to the inverse transformation introduced in the previous section, it is possible to manage recursively any sequence of references between origins.

Other ways to express transformations exist, in particular when the relative position between two frames is expressed with respect to a third reference frame. These alternative ways are not discussed here being not required for the purpose of this work.

2.3 Rotations Representation

Rotation matrices and their properties were briefly introduced in Section 2.2.2. Although rotation matrices are very useful in performing rotations, they are not the simplest notation to code a rotation in 3D space. A rotation in 3D space needs to specify only 3 degrees of freedom (*DoF*), while a rotation matrix contains 9 entries related by mathematical properties (e.g., orthonormality, unit vectors, etc.). Compact notations exist and some of them, with their properties and useful formulas for conversion, are presented in this section; for a more complete analysis refer to [23] and [86].

2.3.1 Euler Angles

Any orientation in 3D space can be reached by performing successive rotations around three different axis. The three angles of rotation around the axes are referred as *Euler angles*. It is worth to notice that 24 different combinations of successive rotations lead to an orientation (e.g., rotate in order around X,Y,Z axes, or X,Z,Y axes, etc.). One of the most commonly used rotation sequence is:

1. Rotate around Z axis (*Yaw* or *heading* angle, ψ)
2. Rotate around the resultant Y' axis (*Pitch* or *elevation*, θ)
3. Rotate around the resultant X'' axis (*Roll* or *bank*, ϕ)

Notice that *Yaw*, *Pitch* and *Roll* are normally used to name angles in the three aircraft principal axes, but they are not equivalent to this convention for rotation representation.

The three elementary rotations around Z, Y and X axes are rotation matrices themselves:

$$\mathbf{R}_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.28}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.29}$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \tag{2.30}$$

and their effects are shown in Figure 2.5; the combination of them gives:

$$\mathbf{R} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{2.31}$$

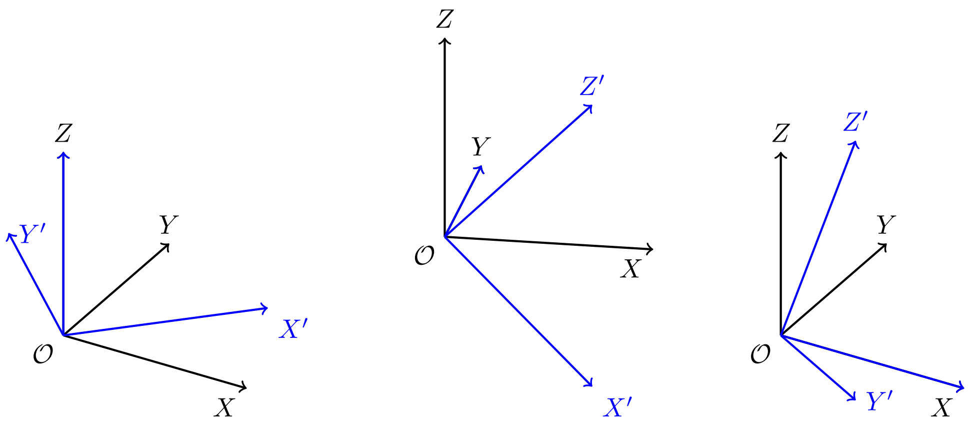
With this convention the extraction of Euler angles from a rotation matrix is done by:

$$\psi = \mathsf{atan2}\left(\mathbf{R}_{21}, \mathbf{R}_{11}\right), \tag{2.32}$$

$$\theta = \mathsf{atan2}\left(-\mathbf{R}_{31}, \sqrt{\mathbf{R}_{32}^2 + \mathbf{R}_{33}^2}\right), \tag{2.33}$$

$$\phi = \mathsf{atan2}\left(\mathbf{R}_{32}, \mathbf{R}_{33}\right). \tag{2.34}$$

The Euler angles convention provides a minimal representation for rotations, i.e., they use exactly 3 parameters to represent the 3 DoF of rotation, but they present discontinuities. Moreover, the composition of rotations is not trivial and does not correspond to the sum of Euler angles. A good choice for the composition of rotation represented by Eluler angles, is to convert Euler angles to rotation matrices, compose them and then extract the resulting Euler angles.



(a) Rotation around Z axis (b) Rotation around Y axis (c) Rotation around X axis

Figure 2.5: *Elementary rotation around a single axis. Original axis X, Y, Z are in black, rotated axis X', Y', Z' are in blue.*

2.3.2 Axis/Angle

A generic rotation could be expressed in terms of a rotation by an angle θ around a specific axis. The axis represents a direction and it is coded by a unit vector $\overline{\mathbf{u}}$. The identity rotation has many infinitely representations: it is a rotation by $0°$ around any axis, i.e., around any unit vector. With this representation, a rotation is expressed with 4 parameters, so it is not minimal. A common choice to obtain a minimal representation from it, called *Rodrigues parameters* or *rotation vector*, is to use a 3-element vector $\Phi = \theta\overline{\mathbf{u}}$. Thus, $\theta = \|\Phi\|$ and $\overline{\mathbf{u}} = \frac{\Phi}{\|\Phi\|}$.

The conversion from a rotation vector to a rotation matrix is given by

$$\mathbf{R}(\Phi) = e^{[\Phi]_\times}. \tag{2.35}$$

The Taylor expansion of this equation, for the properties of the power of cross product matrix (see Section 2.1.4), leads to a closed form, known as the *Rodrigues formula*:

$$\mathbf{R}(\Phi) = \mathbf{I} + \sin\theta \, [\overline{\mathbf{u}}]_\times + (1 - \cos\theta) \, [\overline{\mathbf{u}}]_\times^2 \tag{2.36}$$

Alike for Euler angles, the composition of rotations expressed in the Rodrigues parameters form can not be performed by adding the two rotation vectors, thus the composition can be done by using the equivalent rotation matrices and by converting the resulting matrix.

Rodrigues parameters suffers from a discontinuity at $180°$ (π radians): each vector Φ such that $\|\Phi\| = \pi$ represents the same rotation of $-\Phi$.

2.3.3 Gibbs Vector and Modified Rodrigues Parameters

Two evolutions of the Rodrigues parameters notation are the *Gibbs vector* and the *Modified Rodrigues Parameters (MRP)*.

Starting from the Axis/Angle formulation, the Gibbs vector is defined as

$$\mathbf{g} = \tan\left(\frac{\theta}{2}\right)\overline{\mathbf{u}}, \tag{2.37}$$

while MRP are defined as

$$\mathbf{p} = \tan\left(\frac{\theta}{4}\right)\overline{\mathbf{u}}. \tag{2.38}$$

Gibbs vector cannot represent a 180° rotation, while MRP present, alike Rodrigues parameters, discontinuous jumps in the parameter space when rotation grows.

2.3.4 Quaternions

Quaternions were originally introduced by Sir William R. Hamilton in early 1840's [37] [38] and he soon observed that they could be adopted in 3D geometry operations. A quaternion $\mathbf{q}$ can be thought of as either

- a vector with four components: $[\mathbf{q}_w, \mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z]^T$;
- a scalar plus a vector with three components: $[\mathbf{q}_w, \mathbf{q}_\mathbf{v}^T]^T$;
- a complex number with three different "imaginary" parts: $\mathbf{q}_w + \mathbf{q}_x\,\overline{\mathbf{i}} + \mathbf{q}_y\,\overline{\mathbf{j}} + \mathbf{q}_z\,\overline{\mathbf{k}}$.

Quaternions can be composed by quaternion products operator $\otimes$. To define these operations, basic rules of quaternions multiplications are defined as:

$$\overline{\mathbf{i}}^2 = \overline{\mathbf{j}}^2 = \overline{\mathbf{k}}^2 = \overline{\mathbf{i}}\,\overline{\mathbf{j}}\,\overline{\mathbf{k}} = -1, \tag{2.39}$$

$$\overline{\mathbf{i}}\,\overline{\mathbf{j}} = -\overline{\mathbf{j}}\,\overline{\mathbf{i}} = \overline{\mathbf{k}}, \quad \overline{\mathbf{j}}\,\overline{\mathbf{k}} = -\overline{\mathbf{k}}\,\overline{\mathbf{j}} = \overline{\mathbf{i}}, \quad \overline{\mathbf{k}}\,\overline{\mathbf{i}} = -\overline{\mathbf{i}}\,\overline{\mathbf{k}} = \overline{\mathbf{j}}; \tag{2.40}$$

then, the product of quaternions $\mathbf{q} \otimes \mathbf{p}$ could be defined as

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{Q}(\mathbf{q})\mathbf{p} \tag{2.41}$$

$$= \begin{bmatrix} \mathbf{q}_w & -\mathbf{q}_x & -\mathbf{q}_y & -\mathbf{q}_z \\ \mathbf{q}_x & \mathbf{q}_w & -\mathbf{q}_z & \mathbf{q}_y \\ \mathbf{q}_y & \mathbf{q}_z & \mathbf{q}_w & -\mathbf{q}_x \\ \mathbf{q}_z & -\mathbf{q}_y & \mathbf{q}_x & \mathbf{q}_w \end{bmatrix}\mathbf{p} \tag{2.42}$$

$$\tag{2.43}$$

$$= \left(\mathbf{q}_w\mathbf{I} + \begin{bmatrix} 0 & -\mathbf{q}_\mathbf{v}^T \\ \mathbf{q}_\mathbf{v} & [\mathbf{q}_\mathbf{v}]_\times \end{bmatrix}\right)\mathbf{p}, \tag{2.44}$$

or

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{Q}^+(\mathbf{p})\mathbf{q} \tag{2.45}$$

$$= \begin{bmatrix} \mathbf{p}_w & -\mathbf{p}_x & -\mathbf{p}_y & -\mathbf{p}_z \\ \mathbf{p}_x & -\mathbf{p}_w & \mathbf{p}_z & -\mathbf{p}_y \\ \mathbf{p}_y & -\mathbf{p}_z & -\mathbf{p}_w & \mathbf{p}_x \\ \mathbf{p}_z & \mathbf{p}_y & -\mathbf{p}_x & -\mathbf{p}_w \end{bmatrix}\mathbf{q} \tag{2.46}$$

$$= \left(\mathbf{p}_w\mathbf{I} + \begin{bmatrix} 0 & -\mathbf{p}_\mathbf{v}^T \\ \mathbf{p}_\mathbf{v} & -[\mathbf{p}_\mathbf{v}]_\times \end{bmatrix}\right)\mathbf{q}. \tag{2.47}$$

The identity quaternion is $\mathbf{q} = [1, 0, 0, 0]^T$. The conjugate of a quaternion is $\mathbf{q}^* = \left[\mathbf{q}_w, \; -\mathbf{q}_\mathbf{v}^T\right]^T$, the norm of a quaternion is $\|\mathbf{q}\| = \sqrt{\mathbf{q}_w^2 + \mathbf{q}_x^2 + \mathbf{q}_y^2 + \mathbf{q}_z^2} \equiv \sqrt{\mathbf{q}\mathbf{q}^*}$; if $\|\mathbf{q}\| = 1$ the quaternion is a *unit quaternion* or it is said to be *normalized*.

A rotation in the the Axis/Angle formulation (θ, $\overline{\mathbf{u}}$) can be represented using a quaternion as

$$\mathbf{q} = \left[\cos\left(\frac{\theta}{2}\right), \quad \sin\left(\frac{\theta}{2}\right)\overline{\mathbf{u}}^T\right]^T . \tag{2.48}$$

Thus, a rotation can be represented by a *unit quaternion*.

The unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation: a rotation by angle θ around the axis $\overline{\mathbf{u}}$ can be expressed as a rotation by $-\theta$ angle around the axis $-\overline{\mathbf{u}}$. Similarly, the inverse of a rotation represented by $\mathbf{q}$ is $\mathbf{q}^*$: this comes from the fact that a rotation by θ around $\overline{\mathbf{u}}$ is "undone" by a rotation of θ around $-\overline{\mathbf{u}}$.

The quaternion $\mathbf{q}$ is converted into a rotation matrix thanks to

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} \mathbf{q}_w^2 + \mathbf{q}_x^2 - \mathbf{q}_y^2 - \mathbf{q}_z^2 & 2\left(\mathbf{q}_x\mathbf{q}_y - \mathbf{q}_w\mathbf{q}_z\right) & 2\left(\mathbf{q}_x\mathbf{q}_z + \mathbf{q}_w\mathbf{q}_y\right) \\ 2\left(\mathbf{q}_x\mathbf{q}_y + \mathbf{q}_w\mathbf{q}_z\right) & \mathbf{q}_w^2 - \mathbf{q}_x^2 + \mathbf{q}_y^2 - \mathbf{q}_z^2 & 2\left(\mathbf{q}_y\mathbf{q}_z - \mathbf{q}_w\mathbf{q}_x\right) \\ 2\left(\mathbf{q}_x\mathbf{q}_z - \mathbf{q}_w\mathbf{q}_y\right) & 2\left(\mathbf{q}_y\mathbf{q}_z + \mathbf{q}_w\mathbf{q}_x\right) & \mathbf{q}_w^2 - \mathbf{q}_x^2 - \mathbf{q}_y^2 + \mathbf{q}_z^2 \end{bmatrix}, \tag{2.49}$$

and to apply a rotation to a vector $\mathbf{v}$ there are two possible ways

(a) Convert the quaternion to a rotation matrix and use Equation 2.16:

$$\mathbf{v}' = \mathbf{R}(\mathbf{q})\,\mathbf{v}; \tag{2.50}$$

(b) Code the vector in the imaginary part of a quaternion ($\mathbf{v}_\mathbf{q} = [0, \mathbf{v}^T]^T$) and apply

$$\mathbf{v}' = \mathbf{q} \otimes \mathbf{v}_\mathbf{q} \otimes \mathbf{q}^*. \tag{2.51}$$

Thanks to the quaternion multiplication operation and to the conjugation, quaternions can be used to represent frame relations. Considering two origins $\mathcal{O}$ and $\mathcal{W}$, the transformation $\mathbf{T}_\mathcal{O}^\mathcal{W}$ is represented by the translation $\mathbf{t}_\mathcal{O}^\mathcal{W}$ and the rotation $\mathbf{q}_\mathcal{O}^\mathcal{W}$. Transformation inversion is done by

$$\mathbf{q}_\mathcal{W}^\mathcal{O} = \mathbf{q}_\mathcal{O}^{\mathcal{W}*} \tag{2.52}$$

$$\mathbf{t}_\mathcal{W}^\mathcal{O} = -\mathbf{R}\left(\mathbf{q}_\mathcal{O}^\mathcal{W}\right)^T \mathbf{t}_\mathcal{O}^\mathcal{W}. \tag{2.53}$$

Notice that $\mathbf{R}\left(\mathbf{q}_\mathcal{O}^\mathcal{W}\right)^T = \mathbf{R}\left(\mathbf{q}_\mathcal{O}^{\mathcal{W}*}\right)$.

It can be verified that the transformation in a 3D rotation matrix of the product of two quaternions is equivalent to the product of the two matrices generated by the quaternions, i.e., it is possible to write the relation

$$\mathbf{R}\left(\mathbf{q}_\mathcal{S}^\mathcal{W}\right) = \mathbf{R}\left(\mathbf{q}_\mathcal{O}^\mathcal{W}\right)\mathbf{R}\left(\mathbf{q}_\mathcal{S}^\mathcal{O}\right) = \mathbf{R}\left(\mathbf{q}_\mathcal{O}^\mathcal{W} \otimes \mathbf{q}_\mathcal{S}^\mathcal{O}\right) . \tag{2.54}$$

2.3.5 A Note on Small Rotations

Consider the orientation of a reference system $\mathcal{O}$ with respect to $\mathcal{W}$ composed by the rotation of a intermediate frame $\mathcal{O}'$ with respect to $\mathcal{W}$ and the rotation of $\mathcal{O}$ with respect to $\mathcal{O}'$. Representing both by rotation matrices and quaternions we have:

$$\begin{aligned}\mathbf{R}_{\mathcal{O}}^{\mathcal{W}} &= \mathbf{R}_{\mathcal{O}'}^{\mathcal{W}} \otimes \mathbf{R}_{\mathcal{O}}^{\mathcal{O}'}, &(2.55)\\ \mathbf{q}_{\mathcal{O}}^{\mathcal{W}} &= \mathbf{q}_{\mathcal{O}'}^{\mathcal{W}} \otimes \mathbf{q}_{\mathcal{O}}^{\mathcal{O}'}. &(2.56)\end{aligned}$$

Now, consider that the second transformation is a small perturbation in the local frame and name these terms respectively as $\mathbf{R}_{\mathcal{O}}^{\mathcal{O}'} = \Delta\mathbf{R}$ and $\mathbf{q}_{\mathcal{O}}^{\mathcal{O}'} = \Delta\mathbf{q}$.

Considering a small rotation vector $\mathbf{\Delta\Phi}$, the first order Taylor expansion of Equation 2.35 and Equation 2.48 gives respectively

$$\begin{aligned}\Delta\mathbf{R} &= \mathbf{I} + [\Delta\Phi]_\times + O\left(\|\Delta\Phi\|^2\right), &(2.57)\\ \Delta\mathbf{q} &= \begin{bmatrix}1\\ \frac{\Delta\Phi}{2}\end{bmatrix} + O\left(\|\Delta\Phi\|^2\right). &(2.58)\end{aligned}$$

2.3.6 Time Derivatives

Consider $\mathbf{q}(t)$ the orientation of a body at time t and $\mathbf{q}(t+\Delta t)$ its orientation after a small Δt interval of time. Being $\omega(t) = \lim_{\Delta t\to 0}\frac{\Delta\Phi}{\Delta t}$ the angular rate vector in body frame expressed in terms of a small rotation $\Delta\Phi$, it is possible to apply the difference quotient to obtain the derivative in time of the body orientation:

$$\dot{\mathbf{q}}(t) = \lim_{\Delta t\to 0}\frac{\mathbf{q}(t+\Delta t) - \mathbf{q}(t)}{\Delta t}. \quad (2.59)$$

We can perform these rewritings:

$$\begin{aligned}\mathbf{q}(t+\Delta t) &= \mathbf{q}(t)\otimes\Delta\mathbf{q} &(2.60)\\ &= \mathbf{q}(t)\otimes\left(\begin{bmatrix}1\\ \frac{\Delta\Phi}{2}\end{bmatrix} + O\left(\|\Delta\Phi\|^2\right)\right) &(2.61)\\ &\simeq \frac{1}{2}\mathbf{Q}^+\left(\Delta\Phi\right)\mathbf{q}(t) &(2.62)\\ &= \left(\mathbf{I} + \frac{1}{2}\begin{bmatrix}0 & -\Delta\Phi^T\\ \Delta\Phi & -[\Delta\Phi]_\times\end{bmatrix}\right)\mathbf{q}(t), &(2.63)\end{aligned}$$

where the second order term has been discarded. Thus, the difference quotient can be rewritten as

$$\dot{\mathbf{q}}(t) \simeq \lim_{\Delta t \to 0} \frac{\left(\mathbf{I} + \frac{1}{2}\begin{bmatrix} 0 & -\Delta\Phi^T \\ \Delta\Phi & -[\Delta\Phi]_\times \end{bmatrix}\right)\mathbf{q}(t) - \mathbf{q}(t)}{\Delta t} \tag{2.64}$$

$$= \frac{1}{2}\begin{bmatrix} 0 & -\omega(t)^T \\ \omega(t) & -[\omega(t)]_\times \end{bmatrix}\mathbf{q}(t) \tag{2.65}$$

$$= \frac{1}{2}\mathbf{q}(t) \otimes \begin{bmatrix} 0 \\ \omega(t) \end{bmatrix}. \tag{2.66}$$

Following the same approach, when orientation is represented by a rotation matrix $\mathbf{R}(t)$, the time derivative results in:

$$\dot{\mathbf{R}} = \mathbf{R}\,[\omega(t)]_\times\,. \tag{2.67}$$

2.3.7 Discussion on Rotation Representation

The comparison of the proposed rotation representations can be faced by different points of view. Considering the required storage space, we can easily observe that Rotation matrices use 9 elements, quaternions and the native axis/angle representations use 4 elements, while others representations (the rotation vector, the Gibbs vector, the Modified Rodrigues parameters and the Euler angles) use 3 elements. The latter are minimal, since they represent the three degrees of freedom with three elements, while the former are not minimal.

In [30] a detailed analysis on the computational cost of common operations on transformations, such that transformations of points, transformation compositions and transformation inversions, is performed. It reveals that the use of quaternions or rotation matrix representation is essentially equivalent. This implies that the computational cost for the same operations performed with minimal representations, such that Euler angles or Gibbs vector, is higher, since the representation have to be converted in quaternions or rotation matrix to be properly composed, adding a significant overhead. A further consideration stated in [30] considers the computational cost of the *normalization* of rotation representations. Being both quaternion and rotation matrices overparametrized, they are subject to some constraints. In particular, quaternions have to be unitary and rotation matrices have to be orthogonal with unitary magnitude. These constraints may results violated as repeated calculations, such that compositions or inversions, are performed, due to round-off errors. Thus, it is needed to normalize the representations. Quaternions can be easily normalized by enforcing the unitary module with $\overline{\mathbf{q}} = \frac{\mathbf{q}}{\|\mathbf{q}\|}$. The normalization of rotation matrices is more complex and can be achieved with several approaches. In [30] a procedure is detailed and the analysis of the computational complexity shows a clear advantage for the quaternion representation, which results more convenient both in terms of spatial complexity and computational complexity.

Beside computational complexity, quaternions represent a convenient way of representing rotations for others accessory properties. Quaternions can be easily converted to and from convenient and “human ’readable” minimal representations, such that Euler angles or Axis/Angle representations. Quaternions are easily interpolated in a sequence of smooth rotations between an initial and a final rotation, while this is not easily achieved with other representations. Quaternion representation does not suffer of singularities, apart from the antipodal equality between $\mathbf{q}$ and $-\mathbf{q}$, while it is topologically impossible to have a global 3-dimensional parametrization without singular points for the rotation group, as stated in [99].

CHAPTER 3

Computer Vision Introduction

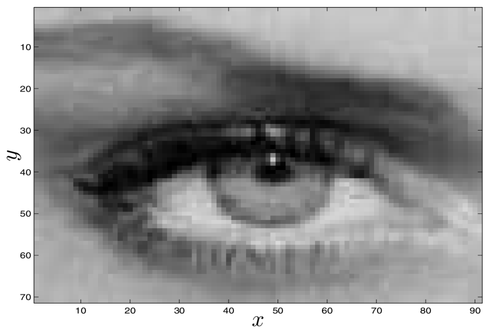
This chapter presents a review of the image formation process focusing on standard *perspective cameras*. The *thin lenses* model is introduced to derive the *pin-hole* geometrical model of point projection and the algorithms for the compensation of lens distortion. Moreover, a brief review of salient features that can be identified in images is presented as a useful reference for the system presented later on this thesis, focusing on salient point detection and feature tracking systems.

3.1 Perspective Camera and Image Formation

A camera is a sensor that is able to perceive light rays and to record the images formed by them on a sensible surface. The term camera comes from the Latin terms *camera obscura* ("dark chamber"), an early system that allowed to project images. Standard cameras perceive light in the visible spectrum, but other sensors exist which perceive different portions of the electromagnetic spectrum (e.g., *infrared* cameras). Modern cameras record images on a CCD (*charge-coupled device*) or CMOS (*Complementary Metal Oxide Semiconductor*) sensor in order to produce *digital images*.

188	186	188	187	168	130	101	99	110	113	112	107	117	140	153	153	156	158	156	153
189	189	188	181	163	135	109	104	113	113	110	109	117	134	147	152	156	163	160	156
190	190	188	176	159	139	115	106	114	123	114	111	119	130	141	154	165	160	156	151
190	188	188	175	158	139	114	103	113	126	112	113	127	133	137	151	165	156	152	145
191	185	189	177	158	138	110	99	112	119	107	115	137	140	135	144	157	163	158	150
193	183	178	164	148	134	118	112	119	117	118	106	122	139	140	152	154	160	155	147
185	181	178	165	149	135	121	116	124	120	122	109	123	139	141	154	156	159	154	147
175	176	176	163	145	131	120	118	125	123	125	112	124	139	142	155	158	158	155	148
170	170	172	159	137	123	116	114	119	122	126	113	123	137	141	156	158	159	157	150
171	171	173	157	131	119	116	113	114	118	125	113	122	135	140	155	156	160	160	152
174	175	176	156	128	120	121	118	113	112	123	114	122	135	141	155	155	158	159	152
176	174	174	151	123	119	126	121	112	108	122	115	123	137	143	156	155	152	155	150
175	169	168	144	117	117	127	122	109	106	122	116	125	139	145	158	156	147	152	148
179	179	180	155	127	121	118	109	107	113	125	133	130	129	139	153	161	148	155	157
176	183	181	153	122	115	113	106	105	109	123	132	131	131	140	151	157	149	156	159
180	181	177	147	115	110	111	107	107	105	120	132	133	133	141	150	154	148	155	157
181	174	170	141	113	111	115	112	113	105	119	130	132	134	144	153	156	148	152	151
180	172	168	140	114	114	118	113	112	107	119	128	130	134	146	157	162	153	153	148
186	176	171	142	114	114	116	110	108	104	116	125	128	134	148	161	165	159	157	149
185	178	171	138	109	110	114	110	109	97	110	121	127	136	150	160	163	158	156	150

(a) Intensity matrix



(b) Gray scale

Figure 3.1: *A digital image, represented as (a) an intensity matrix, (b) with a gray scale mapping.*

3.1.1 Image Formation

An ideal, gray levels, image is a mapping between a subset of the two dimensional space, representing image coordinates, and a real value representing the light *intensity*:

$$\mathbf{I} : \ \Omega \in \mathbb{R}^2 \ \rightarrow \ \mathbb{R}. \tag{3.1}$$

Color images are represented by three different mappings ($\mathbf{I}_R$, $\mathbf{I}_G$, $\mathbf{I}_B$), each of them representing the intensity of a color among *Red*, *Green* and *Blue*. Different mappings color are possible, e.g., CMYK (Cyan, Magenta, Yellow and blacK), HSL (Hue, Saturation and Lightness) and HSV (Hue, Saturation and Value) but they are not useful for this work.

Real images are represented as a mapping in the discrete domain, both for the image coordinates and for the intensity:

$$\mathbf{I} : \ \Omega \in \mathbb{N}^2 \ \rightarrow \ \Lambda \in \mathbb{N}. \tag{3.2}$$

A pixel (*px*) is the unitary element of the image, i.e., the smallest addressable element in an image and it collects the light integrated both in space (i.e., over the pixel area) and in time (i.e., during the exposure time). Thus, an image is a bi-dimensional array of intensities; $\mathbf{I}(y, x)$ is the intensity of the *pixel* of coordinates $\boldsymbol{p} = [x, y]$. Notice that, when representing images, a common convention is to use the Y axis pointing down from the top left corner of the image; the X axis pointing right and Z follow the right hand convention. An example of an image represented respectively as intensity matrix or mapping intensity on a gray scale is shown in Figure 3.1(a) and Figure 3.1(b)

More than the sensor, the principal actors of the image formation process are a *lens* or a set of lenses, an *aperture* and a *shutter*. The role of the lens (or the set of lenses) is to "direct" the light coming from the environment, i.e., imposing a controlled change in the direction of propagation of light rays, which can be performed by means of diffraction, refraction, and reflection. Rays of light are then "selected" by a small aperture. The shutter is a component that triggers the integration of light for a certain amount of time on the camera sensor. For further details on image formation process the reader can refers to [29] and [56].

3.1.2 Thin Lenses

The role of lenses, i.e., the variation of light direction by means of diffraction, refraction, and reflection, needs to be simplified in order to allow a simple mathematical treatment. In particular, the simplest model is the *thin lenses* model, that explain the ray propagation geometry. With reference to the Figure 3.2, the model is defined by the *optical axis* Z, i.e., the axis normal to the lens with its origin $\mathcal{O}$ located at the intersection with the lens symmetry plane (*optical center*). Considering a lens which surfaces are part of spheres forming a convex shape, all parallel rays are deflected by lens ensuring that they converge on a point at distance f (*focal length*) from the optical center. Focal length depends on lens characteristics, but no further details are given here; the interested reader can refer to [40].

The thin lenses model is a suitable approximation when the thickness of the lens d is negligible when compared to f, that makes this properties valid: rays through $\mathcal{O}$ are undeflected from the lens. From this follows that rays coming from a 3D point $\mathbf{P}^{(\mathcal{O})} = [x, y, z]$ converge at distance r from the optical center following the Fresnel law:

$$\frac{1}{f} = \frac{1}{z} + \frac{1}{r}. \tag{3.3}$$

Notice that $z \to \infty$ implies $r \to f$. The proof of the Fresnel law is illustrated in Figure 3.2 and comes from the similarities of couple of triangles expressed by

$$\frac{h}{y} = \frac{r-f}{f}, \tag{3.4}$$

$$\frac{h}{y} = \frac{r}{z}. \tag{3.5}$$

The Fresnel law states that points at different distance from the optical center converge at a different position. This implies that if the image plane π_I (i.e. the camera sensor) is at a distance l from the optical center, a point is projected to a *blurring circle* C, as illustrated in Figure 3.3. To minimize this effect a small aperture of diameter a is used, in order to avoid the passage of lights coming from the same point with high direction differences. A point is said to be *focused* if the diameter of C ($\Phi(C) = \frac{a\,(l-r)}{r}$) is smaller than a pixel. Thus, it is possible to identify a range $[z_1, z_2]$ (*depth of field*) for distances that ensures that points are properly focused. The choice of a proper aperture has a trade-off: the smaller is the aperture, the larger is the depth of field range but a smaller amount of light arrives on the camera sensor; shooting time interval needs to be longer and this could cause a motion blur if the shotted scene is not static (or the camera is moving). Obviously a bigger aperture prevents good images if the depth range of points is bigger than depth of field.

Considering a sufficiently small aperture and a distance $z \gg f$ (that implies $r \simeq f$), the thin lens model could be simplified into the *pin-hole* model: for each point only a single ray passes from the optical center and it is focused at distance f. Thus, placing the image plane $\pi_{\mathbf{I}}$ at distance f allows simple relations between points and theirs projection on the image plane.

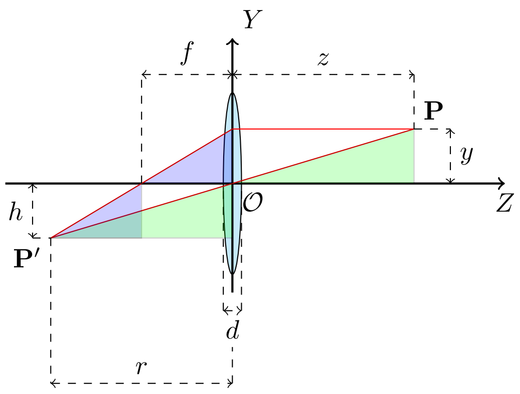


Figure 3.2: *Thin lens model and Fresnel Law in 2D side view. Two rays starting from 3D $\mathbf{P}^{(\mathcal{O})} = [x, y, z]$ are considered: the first cross the optical center $\mathcal{O}$ thus it is undeflected; the second is parallel to the optical axis, thus it cross the optical axis in $-f$. Similarities between triangles allow to write (1) $\frac{h}{y} = \frac{r-f}{f}$ for blue triangles and (2) $\frac{h}{y} = \frac{r}{z}$ for green triangles.*

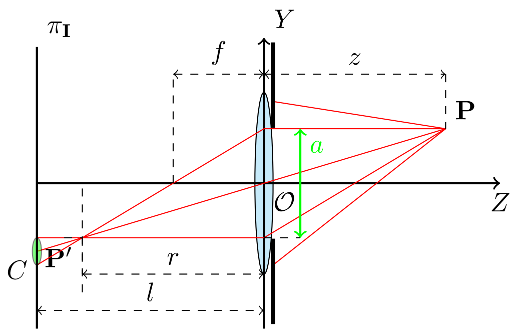


Figure 3.3: *Effects of aperture a in point projection: formation of the blurring circle C. The image plane $\pi_\mathbf{I}$ is placed at distance l from the optical center (O), but rays from point $\mathbf{P}$ converge at distance r (all points with distance z converge at distance r).*

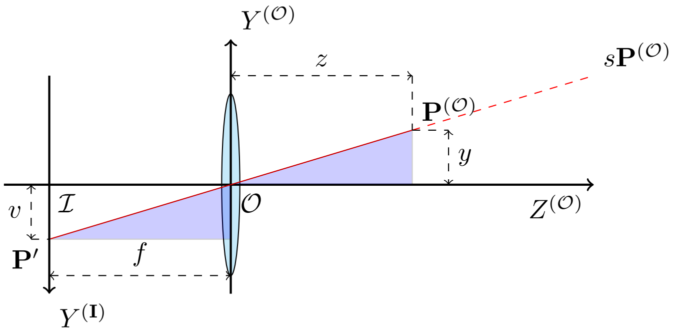


Figure 3.4: *Pin-hole camera model with projection equations for the plane $Y - Z$. Considering similarities between the blue triangles allows to write $\mathbf{P}' = \left[f\frac{x}{z}, f\frac{y}{z}\right]$.*

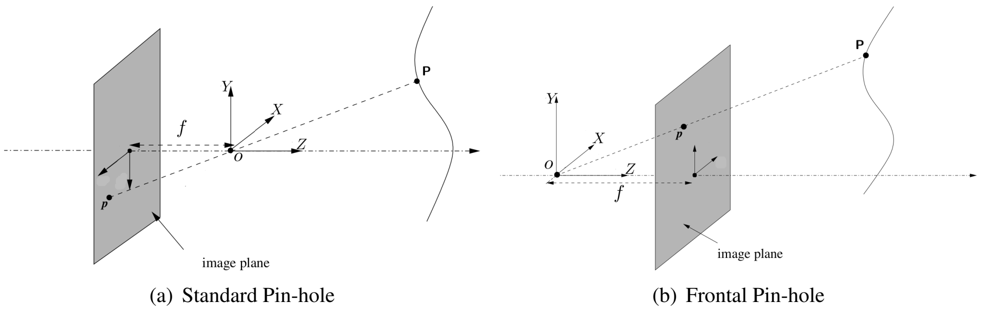


Figure 3.5: *The pin hole model with (a) the image plane behind the optical center (b) the image plane ahead the optical center.*

3.1.3 The Pin-hole Model

Considering a 3D point $\mathbf{P}^{(\mathcal{O})} = [x, y, z]^T$ (in Cartesian coordinate) it will be projected in the 2D point in the image plane at $\boldsymbol{p} = \left[f\frac{x}{z}, f\frac{y}{z}\right]$. This could be easily proven with triangles similarities showed in Figure 3.4 for the plane $Y - Z$. Indeed, all the points that lie on the same line that join $\mathbf{O}$ and $\mathbf{P}^{(\mathcal{O})}$, i.e., points $s\mathbf{P}^{(\mathcal{O})}$ with $s \neq 0$ project on the same image point $\boldsymbol{p}$.

The image plane is physically placed behind the optical center, but from a mathematical point of view it is equivalent to consider it placed ahead the optical center, with an obvious change of direction of the axis of the image plane. This is summarized in Figure 3.5.

A convenient convention is to introduce a *normalized image plane* π_1 at distance 1 from the optical center. This allow to treat a generic camera as an ideal camera with focal length equal to 1. Thus, given a 3D point $\mathbf{P}^{(\mathcal{O})} = [x, y, z]^T$, its 2D coordinates on the normalized image plane are

$$\boldsymbol{p} = \left[\tfrac{x}{z}, \tfrac{y}{z}\right]^T, \tag{3.6}$$

while, a given a point $[u, v]^T$ in the image is equivalent to $\left[\frac{u}{f}, \frac{v}{f}\right]^T$ on the normalized image plane.

3.1.4 Homogeneous Camera Matrix

The projection equations introduced in the previous section need some refinement to model a real camera. The first issue is due to the reference system on the image plane: it is placed in the middle of the image, i.e., its origin is placed on the intersection between the optical axis and the image plane. This is not true in the reality: an image is represented by a matrix, thus origin is in the top-left corner. Moreover, image coordinate system unit is the pixel, while 3D points are expressed with metric coordinates. Finally, pixels are not guaranteed to be squared, while they could be rectangles or parallelograms for structural reasons, especially in old cameras.

Conversion between metric coordinates and pixels (including the rectangular pixel case) can be easily achieved by a non uniform scaling; let consider d_x, d_y the pixel dimensions, then $f_y = \frac{f}{d_x}$ and $f_y = \frac{f}{d_y}$ become the focal lengths in pixel that convert points from metric coordinates to pixels. To compact the notation $\mathbf{f} = [f_x, f_y]$ is used.

The translation between the centered image plane reference system and the top-left reference system of digital images could be easily treated by a translation of $\mathbf{c} = [c_x, c_y]$ pixels, that expresses the position of the central point (i.e., the intersection of the optical axis with the image plane) in pixel coordinates.

The operations to project a 3D point in an image could be coded by a 3×3 homogeneous matrix, named the *camera matrix*:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.7}$$

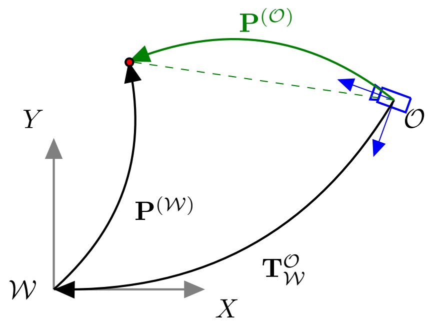
where the factor s allows to compensate pixel with parallelogram shape. The value of s is 0 in almost all cases, so it will not be considered hereafter. $\mathbf{f}$ and $\mathbf{c}$ are usually referred as *intrinsic camera parameters*, because they depends on the camera physical structure of the optical system.

A 3D cartesian point $\mathbf{P}^{(\mathcal{O})} = [x, y, z]^T$ is projected to the homogeneous 2D point $[x', y', \omega']^T$, equivalent to the 2D point on the image $[u, v]^T$ by the following relations:

$$\begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} f_x\, x + c_x\, z \\ f_y\, y + c_y\, z \\ z \end{bmatrix}, \tag{3.8}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{x'}{\omega'} \\ \frac{y'}{\omega'} \end{bmatrix} \equiv \begin{bmatrix} f_x\, \frac{x}{z} + c_x \\ f_y\, \frac{y}{z} + c_y \end{bmatrix}. \tag{3.9}$$

Since all points laying on the same line that join the camera optical center $\mathcal{O}$ and the point $\mathbf{P}^{(\mathcal{O})}$ have the same projection in the image, the projection equation could be



$\mathbf{P}^{(\mathcal{O})}$
Y
$\mathcal{O}$
$\mathbf{P}^{(\mathcal{W})}$
$\mathbf{T}_{\mathcal{W}}^{\mathcal{O}}$
$\mathcal{W}$
X

Figure 3.6: *Projection of point $\mathbf{P}^{(\mathcal{W})}$, expressed in a different reference system than the camera one. Transformation $\mathbf{T}_{\mathcal{W}}^{\mathcal{O}}$ is known, thus $\mathbf{P}^{(\mathcal{O})}$ can be easily computed.*

rewritten in terms of homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} = [\mathbf{K}\ \mathbf{0}]\ \mathbf{P}^{(\mathcal{O})} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}. \tag{3.10}$$

.

Considering a point $\mathbf{P}^{(\mathcal{W})}$ in a different reference system and given the $\mathbf{T}_{\mathcal{W}}^{\mathcal{O}}$ that express the position and orientation of the origin $\mathcal{W}$ with respect to the camera origin $\mathcal{O}$ (see Figure 3.6), the projection of $\mathbf{P}^{(\mathcal{W})}$ on the image is given by:

$$\mathbf{P}^{(\mathcal{O})} = \mathbf{T}_{\mathcal{W}}^{\mathcal{O}} \mathbf{P}^{(\mathcal{W})}, \tag{3.11}$$

$$\begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} = [\mathbf{K}\ \mathbf{0}]\ \mathbf{P}^{(\mathcal{O})}. \tag{3.12}$$

Considering the transformation as an homogeneous matrix

$$\mathbf{T}_{\mathcal{W}}^{\mathcal{O}} = \begin{bmatrix} \mathbf{R}_{\mathcal{W}}^{\mathcal{O}} & \mathbf{t}_{\mathcal{W}}^{\mathcal{O}} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.13}$$

Equation 3.11 and Equation 3.12 gives

$$\begin{bmatrix} x \\ y \\ \omega \end{bmatrix} = \begin{bmatrix} \mathbf{K}\mathbf{R}_{\mathcal{W}}^{\mathcal{O}} & \mathbf{K}\mathbf{t}_{\mathcal{W}}^{\mathcal{O}} \end{bmatrix} \mathbf{P}^{(\mathcal{W})} = \boldsymbol{\pi}\, \mathbf{P}^{(\mathcal{W})}, \tag{3.14}$$

where $\boldsymbol{\pi}$ is the *complete projection matrix*) i.e., a 3×4 matrix that summarize the intrinsic camera matrix and the transformation between the two different reference systems. In particular, $\mathbf{T}_{\mathcal{W}}^{\mathcal{O}}$ express the *extrinsic parameters* of the camera, i.e., parameters that are not dependent from the physical device, but they depends only on the position of points reference system with respect camera reference system.

3.1.5 Interpretation Line

Given a point $\boldsymbol{p} = [u, v]$ on the image it is not possible to know which 3D points has generated it, since projection is surjective: all points lying on the same line starting from the camera optical center $\mathcal{O}$ are projected on the same image point, i.e., a camera is a bearing only sensor and it looses the information about the depth of observed points. From image points it is possible to calculate the *interpretation line*, i.e., the direction of all the points that generates the same image point. This could be done by inverting the Equation 3.8:

$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{f_x} & 0 & \frac{-c_x}{f_x} \\ 0 & \frac{1}{f_y} & \frac{-c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.15)$$

$$\mathbf{d}^{(\mathcal{O})} = \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u-c_x}{f_x} \\ \frac{v-c_y}{f_y} \\ 1 \end{bmatrix}, \quad (3.16)$$

$$\overline{\mathbf{d}}^{(\mathcal{O})} = \frac{\mathbf{d}^{(\mathcal{O})}}{\|\mathbf{d}^{(\mathcal{O})}\|}, \quad (3.17)$$

$$\boldsymbol{p}^{(I)} = \begin{bmatrix} \frac{u-c_x}{f_x} \\ \frac{v-c_y}{f_y} \end{bmatrix}. \quad (3.18)$$

where

- $\mathbf{d}^{(\mathcal{O})}$ is the non-unit direction vector of the interpretation line. Applying it to the origin $\mathcal{O}$ gives the point projection on the normalized image plane, i.e., at distance 1 from the optical center;

- $\overline{\mathbf{d}}^{(\mathcal{O})}$ is the 3D unit vector that represents the direction of the interpretation line starting from $\mathcal{O}$;

- $\boldsymbol{p}^{(I)}$ is the 2D coordinate of the image point $\boldsymbol{p}$ on the normalized image plane.

$\mathbf{P}^{(\mathcal{O})}(\lambda) = \lambda\,\overline{\mathbf{d}}^{(\mathcal{O})}$, $\lambda > 0$ is the parametric interpretation line of $[u, v]$ being λ the distance from $\mathcal{O}$ (see Figure 3.7).

The interpretation line has been expressed in the camera reference system $\mathcal{O}$. To obtain it in a different reference frame, the frame composition rule has to been applied. Given $\mathbf{T}_{\mathcal{W}}^{\mathcal{O}}$ and $\mathbf{P}^{(\mathcal{O})}(\lambda)$ in homogeneous coordinates, $\mathbf{P}^{(\mathcal{W})}(\lambda)$ is:

$$\mathbf{T}_{\mathcal{O}}^{\mathcal{W}} = \begin{bmatrix} {\mathbf{R}_{\mathcal{W}}^{\mathcal{O}}}^T & -{\mathbf{R}_{\mathcal{W}}^{\mathcal{O}}}^T \mathbf{t}_{\mathcal{W}}^{\mathcal{O}} \\ \mathbf{0} & 1 \end{bmatrix}, \quad (3.19)$$

$$\begin{aligned} \mathbf{P}^{(\mathcal{W})}(\lambda) &= \mathbf{T}_{\mathcal{O}}^{\mathcal{W}} \begin{bmatrix} \mathbf{P}^{(\mathcal{O})}(\lambda) \\ 1 \end{bmatrix} & (3.20) \\ &= {\mathbf{R}_{\mathcal{W}}^{\mathcal{O}}}^T \mathbf{P}^{(\mathcal{O})}(\lambda) - {\mathbf{R}_{\mathcal{W}}^{\mathcal{O}}}^T \mathbf{t}_{\mathcal{W}}^{\mathcal{O}}. & (3.21) \end{aligned}$$

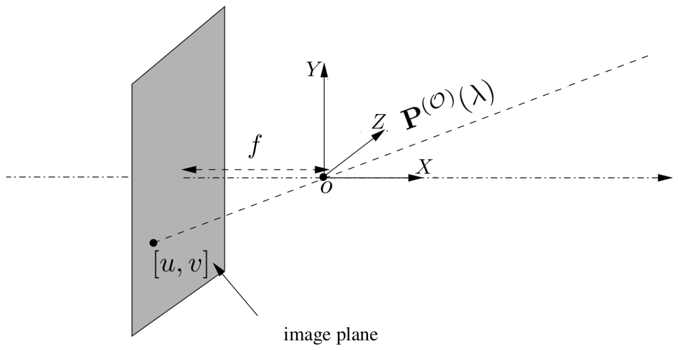


Figure 3.7: *The parametric interpretation line* $\mathbf{P}^{(\mathcal{O})}(\lambda)$ *of image point* $[u, v]$.

3.1.6 Lens Distortion

Almost all real cameras produce images that do not correspond exactly to the ideal projective model due to *distortion* effects. Distortion comes from non-ideal lenses and affects mostly low cost cameras and wide angle optics (i.e., with a short focal length). This would make the pinhole camera model unsuitable unless distortion is compensated by a proper transformation.

Distortion is a non linear effect and can be separated in *tangential* distortion and *radial* distortion. Radial distortion depends on the distance of the considered pixel from the central point, while tangential effects are perpendicular to the line that joins a pixel to the central point. Usually, radial distortion plays a more important role than the tangential one.

Hereafter we consider the polynomial model to models distortion introduced in [7]. Given a point $\boldsymbol{p} = [x, y]^T$ on the normalized image plane (obtained by applying Equation 3.6 to a 3D point or Equation 3.18 to an image point) the distorted point on the normalized image plane is given by

$$r^2 = x^2 + y^2, \tag{3.22}$$

$$\boldsymbol{p}_d = \left(1 + \mathbf{k}_1 r^2 + \mathbf{k}_2 r^4 + \mathbf{k}_3 r^6\right) \boldsymbol{p} + \boldsymbol{d}, \tag{3.23}$$

$$\boldsymbol{d} = \begin{bmatrix} 2\mathbf{t}_1 xy + \mathbf{t}_2 \left(r^2 + 2x^2\right) \\ \mathbf{t}_1 \left(r^2 + 2y^2\right) + 2\mathbf{t}_2 xy \end{bmatrix}, \tag{3.24}$$

where r^2 is the distance from the image center (the center is in $[0, 0]$ in the normalized image plane), $\mathbf{k} = [\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3]^T$ are the coefficient of the radial distortion, modeled by a cubic function of the distance, and $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2]^T$ are the coefficient of the additive tangential distortion part $\boldsymbol{d}$.

On the other side, given a distorted point $\boldsymbol{p} = [u, v]$, i.e. a point on the image plane, it does not exist a closed form for the inverse of Equation 3.23 to remove the distortion. Thus, after the use of Equation 3.18 to obtain the distorted point on the normalized image plane $\boldsymbol{p}_d$, the iterative solution sketched in Algorithm 1 gives the undistorted point $\boldsymbol{p}_u$.

Algorithm 1 $\boldsymbol{p}_u$ = Undistort($\boldsymbol{p}_d$, $\mathbf{k}$, $\mathbf{t}$)

```
1: $\boldsymbol{p}_u \leftarrow \boldsymbol{p}_d$
2: for $i = 1 \rightarrow n$ do
3:     $[x, y] \leftarrow \boldsymbol{p}_u$
4:     $r^2 \leftarrow x^2 + y^2$
5:     $k \leftarrow 1 + \mathbf{k}_1 r^2 + \mathbf{k}_2 r^4 + \mathbf{k}_3 r^6$
6:     $\boldsymbol{d} \leftarrow \begin{bmatrix} 2\mathbf{t}_1 xy + \mathbf{t}_2 \left(r^2 + 2x^2\right) \\ \mathbf{t}_1 \left(r^2 + 2y^2\right) + 2\mathbf{t}_2 xy \end{bmatrix}$
7:     $\boldsymbol{p}_u \leftarrow \frac{\boldsymbol{p}_d - \boldsymbol{d}}{k}$
8: end for
```

3.1.7 Unified Mathematical Notation

The mathematical process described in the previous sections aims to compute how a 3D point, expressed in Homogeneous or Cartesian coordinates, is projected on a 2D Cartesian image point, while the inverse process defines the direction of the viewing ray given the location of a 2D image point.

In particular, the complete projection system of an Homogeneous 3D point $\mathbf{P}_h$ referred to the camera coordinate system, consists in the following steps:

1. convert the homogeneous point $\mathbf{P}_h$ into the Cartesian equivalent $\mathbf{P}$ with Equation 2.14,
2. compute the point on the normalized image plane, i.e., the point $\boldsymbol{p}$ with Equation 3.6,
3. apply distortion to $\boldsymbol{p}$, obtaining $\boldsymbol{p}_d$ with Equation 3.23,
4. apply the projection with Equation 3.9, with $z = 1$, obtaining the true image point $\boldsymbol{p}_I$.

It can be easily noticed that steps 1 and 2 can be reduced to the step 2 only, since the projection on the normalized image plane is not influenced by the scale factor. Moreover, if the original point is in Cartesian coordinates, step 1 has to be skipped.

To compact the notation, in the remaining part of the thesis all these steps will be referenced with the functional notation

$$\boldsymbol{p}_I = K(\mathbf{P}), \tag{3.25}$$

where all the camera parameters ($\mathbf{K}$ camera matrix, lens distortion parameters) are leaved implicit and we do not take care if the source point is expressed in Cartesian or Homogeneous coordinates, leaving to the reader (and to the specific implementation) the choice on the proper steps to apply. When a multi-camera system is involved, a subscript is used to identify a particular camera, e.g., $K_l(\cdot)$ and $K_r(\cdot)$ are commonly used for left and right camera of a stereo rig system.

The inverse process aims to compute the viewing ray direction given a distorted image point $\boldsymbol{p}_d$, involving

1. compute the point on the normalized image plane $\boldsymbol{p}^{(\mathcal{I})}$ with Equation 3.18,
2. compute the undistorted point $\boldsymbol{p}_u$ with Algorithm 1,
3. append a 1 to $\boldsymbol{p}_u$, resulting in $\mathbf{d}^{(\mathcal{O})}$, i.e., the vector that links the camera origin $\mathcal{O}$ to the point the normalized image plane, representing the direction of the interpretation line (see Equation 3.16),
4. in case, compute $\overline{\mathbf{d}}^{(\mathcal{O})}$ with Equation 3.17, i.e., the unit vector direction of the interpretation line.

Steps 1 and 2, i.e., computation of the point on the normalized image plane, will be referred with $K^I(\boldsymbol{p}_d)^{-1}$. When step 3 is performed, i.e., the unnormalized direction vector is computed, the notation will be simply $K(\boldsymbol{p}_d)^{-1}$ while including step 4 too, i.e., computing the unit direction vector, notation will be $\bar{K}(\boldsymbol{p}_d)^{-1}$.

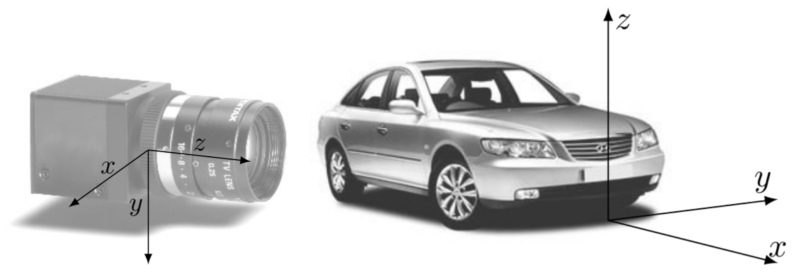
3.1.8 Camera Calibration And Camera Reference System

Intrinsic parameters of the camera are usually estimated by some standard method (among them we can cite [106] and [41], that inspires the useful *Matlab Calibration Toolbox*[1].), since nominal values are not sufficiently precise (e.g., there exists differences due to assembling misalignment, lens productive process and so on). Extrinsic parameters could be estimated too, but this situation is usual only when the camera is fixed (e.g., in a fixed industrial setup or in video surveillance systems). When camera is mounted on a mobile robot or generally it is moving, the complete projection matrix needs to be computed at each time with the current robot position and orientation with respect to a "world" reference frame.

The convention on camera reference frame covers an important role when the transformation between a reference system and the camera has to be computed. With reference to Figure 3.8(a), observing a camera, the Cartesian reference system is placed with the Z axis normal to the the image plane and leaving the camera in the lens direction; the Y axis points down and X points right on the image. This convention is quite different from what is normally used, where Z usually points up while direction of X and Y axes depends on the specific application (e.g., X usually points forward if the reference frame is applied to a mobile robot, as shown in Figure 3.8(b)).

Considering a camera placed on a robot, its orientation with respect to the robot is expressed by the matrix $\mathbf{R}_{\mathcal{O}}^{\mathcal{R}} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$ where $\mathcal{R}$ is the robot reference frame and $\mathcal{O}$ is the camera reference frame. The Z axis of the camera reference frame coincides with the X axis of the robot reference frame, the Y axis of the camera reference frame coincides with the $-Z$ axis of the robot reference frame and the X axis of the camera reference frame coincides with the $-Y$ axis of the robot reference frame.

[1]`http://www.vision.caltech.edu/bouguetj/calib_doc`



(a) Camera Reference System (b) Robot Reference System

Figure 3.8: *The reference system convention (a) for a camera (b) for a standard robotic system.*

3.2 Image Features

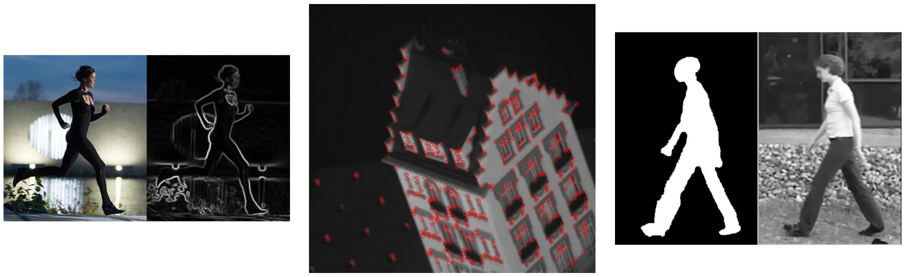
In addition to the geometrical aspects of cameras optical system, the computer vision field aims to give to an artificial system the ability to "understand" the image content. Although there is no a common or exact definition, the terms *feature* is used to refer to an interesting part of the image, where the concept of "interest" strictly depends on the application domain. Often, in real contexts, only small portions of the images are relevant. For the SLAM system that will be described in the next chapters, we are interested in identify 3D points in the world associated to "interesting" features in the 2D projected image.

Consequently, two basic operations need to be defined: the *feature detection* and the *feature matching*. The former examines an image (or a portion of it) to identify the locations where a certain characteristic (i.e., a *feature*) is present; then the feature is stored through a *descriptor* that summarizes in a proper way its characteristics. Feature matching process recognizes a feature in a new image as one of the already observed ones thanks to the stored descriptor. When the matching is performed over time, i.e., on a sequence of images, the process is referred as *feature tracking*.

To the purpose of this work, for which the core operation is feature tracking in image sequences, the interesting features have to satisfy the following properties:

- *Repeatability*: the same feature can be found and matched in several images despite geometric and photometric transformations,
- *Saliency*: each feature has a distinctive *description*,
- *Compactness and efficiency*: a small number (with respect to the number of pixels of the entire image) of features are able to describe a scene,
- *Locality*: if a feature occupies a relatively small area of the image it is more robust to clutter and occlusion.

Some examples of features (see Figure 3.9) are *edges*, i.e., the boundary between different regions, *corners*, i.e., namely intersection between edges but also points with high variation of intensity in all directions and *blobs*, i.e., smooth homogeneous areas that define



(a) Edges (b) Corners (c) Blobs

Figure 3.9: *Some example of features in images.*

regions. To define a complete feature tracking system three fundamental building block are:

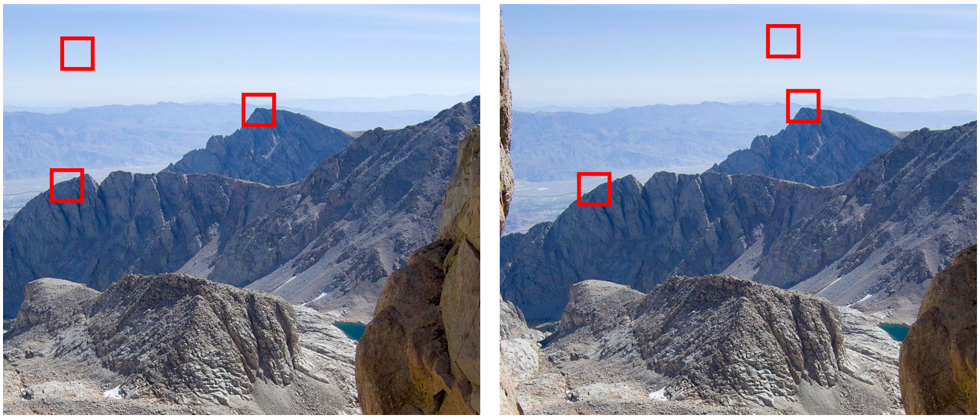
- a *feature detection* algorithm,
- a *feature descriptor*,
- a *feature matching* algorithm.

The latter algorithm is strongly linked with the specific descriptor, because the matching process is build up on the ability to retrieve the stored description of a feature in a successive image. Similarly, the feature detection process influences the matching process, since the detection of distinguishable and descriptive features is a key aspect for successive matching. Consequently, the detection, description and matching/tracking techniques are strictly coupled.

Next sections describe some standard techniques and how they can be used to build a feature tracking system. A good review of feature detectors, descriptors and matchers/-trackers is provided in [101]; other extended reviews and surveys can be found in [84], [64] and [105]. Recent works on this topic have produced interesting techniques that cover all the three aspects of feature tracking. Among them we can cite the *Scale Invariant Feature Transforms (SIFT)* [54] and *Speeded Up Robust Features (SURF)* [4]. These feature detector/descriptors and matchers have the big advantage of being scale and rotation invariant, i.e., they are able to match features that have been scaled and rotated with respect to the descriptor taken at feature initialization. Although this seems a very useful characteristic, SIFT and SURF do not provide an adequate repeatability in feature tracking system applied to SLAM, as shown in [66].

3.2.1 Feature detection

We are interested in detect locally distinguishable features, namely *salient points*. Although we are describing only the feature detection step, we have to reason in broader terms: the choice of a good salient point can increase the distinguishability of a feature. Let consider the image in Figure 3.10(a) and the three features identified by the red squares and consider the portion of image in the square as the descriptor of the feature itself. If



(a) First image (b) Second image

Figure 3.10: *Example of locally distinguishable features and their matching in different images. It is easy to understand that features containing corners (the square located at the peak of the mountain) are more easily recoverable in a different image, while features on edges (the red square on the mountain slope) could be confused with similar areas along the same edge. Homogeneous areas (the red square in the sky) are poorly descriptive.*

we imagine to match them in the second image (Figure 3.10(b)), it is easy to understand that features containing corners (the features containing the mountain peak) have higher saliency with respect to features taken in homogeneous areas (the features containing the part of the blue sky). Features around edges (the feature on the mountain slope) are a bad choice too, because their appears similar along all the edge. Thus, a feature tracking mechanism can perform better if the features to track are chosen where corners are located.

3.2.1.1 Harris And Shi-Tomasi Salient Point Detectors

A good salient point is located where there is an high variation of intensity in all directions. This implies that, in addition to proper corners (edges intersections), good salient points are also isolated points, line endings or points on a curve where the curvature is locally maximal. Without entering in details, two of the most known salient point detector are the Shi-Tomasi [85] and the Harris [39] algorithms. Both algorithms consider the weighted sum of squared differences (SSD) between two patches displaced of (x, y) pixels and covering an area $U \times V$ on the same image $\boldsymbol{I}$, where $U = [u_1 \dots u_n]$ and $V = [v_1 \dots v_n]$ describes a rectangular area by row and column indexes:

$$S(x, y) = \sum_{u \in U} \sum_{v \in V} w(u, v) \left(\boldsymbol{I}(u + x, v + y) - \boldsymbol{I}(x, y) \right)^2 ; \tag{3.26}$$

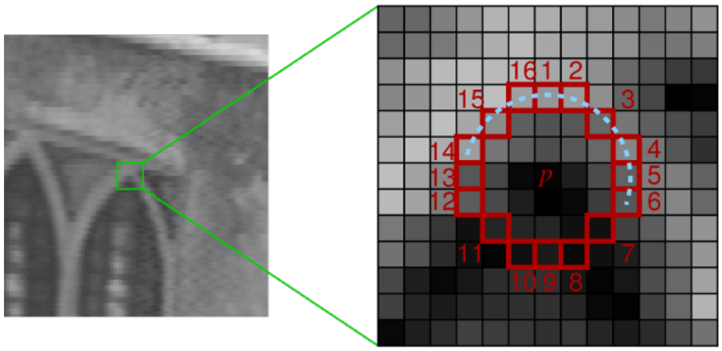


Figure 3.11: *FAST corner detector: pixel on the circumference with radius r are tested.*

where $w(\cdots)$ is a weight function that acts as a smoother on the image. Taylor approximation allows to write the SSD formula in matrix form

$$S(x,y) \simeq \begin{bmatrix} x & y \end{bmatrix} \mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.27}$$

$$\mathbf{A} = \sum_u \sum_v w(u,v) \begin{bmatrix} \boldsymbol{I}_x^2 & \boldsymbol{I}_x\boldsymbol{I}_y \\ \boldsymbol{I}_x\boldsymbol{I}_y & \boldsymbol{I}_y^2 \end{bmatrix} = \begin{bmatrix} \langle \boldsymbol{I}_x^2 \rangle & \langle \boldsymbol{I}_x\boldsymbol{I}_y \rangle \\ \langle \boldsymbol{I}_x\boldsymbol{I}_y \rangle & \langle \boldsymbol{I}_y^2 \rangle \end{bmatrix} \tag{3.28}$$

where the angle brackets stands for the weighted sum.

A salient point is characterized by a large variation of S in all directions. This corresponds to two “large” eigenvalues of the $\mathbf{A}$ matrix. When both eigenvalues of $\mathbf{A}$ are small, the region is homogeneous and it does not contains salient point. When only one of the eigenvalues is significantly different from zero the region contains an edge, since the gradient variation is in a precise direction.

The Shi-Tomasi detector considers a point (x, y) to be a salient point if the smallest eigenvalues of $\mathbf{A}$ is greater than a threshold, while the Harris salient point detector avoids the explicit calculation of eigenvalues by the computation of the metric

$$c_h = \det(\mathbf{A}) - k\,\mathrm{trace}^2(\mathbf{A}). \tag{3.29}$$

and select points which c_h value is over a threshold.

3.2.1.2 FAST

FAST (Features from Accelerated Segment Test), presented in [82] [83], is an alternative approach to corner detection. It tests the pixels on a circumference with radius r around the candidate point $\boldsymbol{p} = (x, y)$ (see Figure 3.11). If n contiguous pixels are all brighter than the candidate point by at least a given threshold t or all darker by t, then the candidate point is considered to be a corner. The test on pixel brightness are conducted in a precise order, trained by a machine learning algorithm, that allows to build short decision tree and to perform feature detection in a computationally efficient way. This test is reported to produce very stable features.

3.2.2 Feature Description and Matching

The description of a feature, that for our purpose is centered on a point identified with one of the proposed salient point detectors, could be achieved in different ways, as listed in [101]. For the aim of this work it is sufficient to consider the description of a features as the part of the image surrounding the salient point, namely a rectangular *patch* of the image, as done in the example of Figure 3.10. To perform feature matching and tracking, once a salient point is identified, the patch around it is saved to allow successive comparison with next images.

3.2.2.1 Template Matching

Template matching is a technique for finding small parts of an image which match a template image, i.e., to find the locations where the similarity between the template and the image is maximized. Lets consider the image $\boldsymbol{I}$ with dimension $W \times H$ and the template image $\boldsymbol{T}$ with dimension $w \times h$, where $w < W$ and $h < H$. Notice that the image $\boldsymbol{I}$ can be a interesting portion of a bigger image where the template matching has to be performed.

The similarity between a $w \times h$ area with top left corner located at x, y coordinates in image $\boldsymbol{I}$ and the template $\boldsymbol{T}$ can be evaluated with the *Normalized Cross Correlation* (NCC) as

$$\boldsymbol{R}_{NCC}(x,y) = \frac{\sum_{x',y'} \left(\boldsymbol{T}(x',y') \cdot \boldsymbol{I}(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} \boldsymbol{T}(x',y')^2 \cdot \sum_{x',y'} \boldsymbol{I}(x+x',y+y')^2}}, \tag{3.30}$$

where

$$x' = [0 \ldots w-1], \tag{3.31}$$
$$y' = [0 \ldots h-1]. \tag{3.32}$$

A different similarity measurement is defined by the *Zero-Mean Normalized Correlation Coefficient* (ZMNCO)

$$\boldsymbol{R}_{ZMNCO}(x,y) = \frac{\sum_{x',y'} \left(\boldsymbol{T}'(x',y') \cdot \boldsymbol{I}'(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} \boldsymbol{T}'(x',y')^2 \cdot \sum_{x',y'} \boldsymbol{I}'(x+x',y+y')^2}}; \tag{3.33}$$

where

$$x' = x'' = [0 \ldots w-1], \tag{3.34}$$
$$y' = x'' = [0 \ldots h-1], \tag{3.35}$$
$$\boldsymbol{I}'(x+x',y+y') = \boldsymbol{I}(x+x',y+y') - \frac{1}{w \cdot h} \sum_{x'',y''} \boldsymbol{I}(x+x'',y+y''), \tag{3.36}$$
$$\boldsymbol{T}'(x',y') = \boldsymbol{T}(x',y') - \frac{1}{w \cdot h} \sum_{x'',y''} \boldsymbol{T}(x'',y''). \tag{3.37}$$

that compute the normalized cross correlation between the image region and the template with their mean removed. Notice that the mean value is not calculated on the entire image

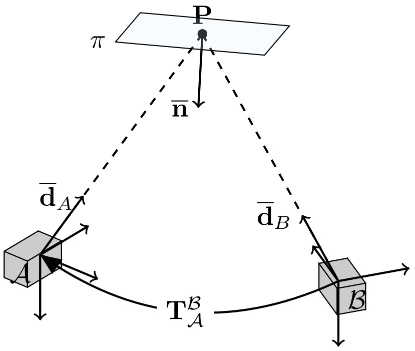

π
$\mathbf{P}$
$\overline{\mathbf{n}}$
$\overline{\mathbf{d}}_A$
$\overline{\mathbf{d}}_B$
$\mathcal{A}$
$\mathcal{B}$
$\mathbf{T}_{\mathcal{A}}^{\mathcal{B}}$

Figure 3.12: *Warping of patches appearance in feature tracking problem*

but only on the area involved in the similarity measure evaluation. Both the similarity measures assume values in the range $[-1, 1]$, where the value 1 corresponds to a perfect match between the image area and the template.

The similarity measure has to be repeated by sliding the template over all the considered image, collecting the results in a matrix $\boldsymbol{R}(x, y)$ with dimension $W-w-1\times H-h-1$. The location in which $\boldsymbol{R}(x, y)$ is maximal correspond to the most suitable location of the template in the image. The template is considered matched at $x^*, y^* = \operatorname{argmax}_{x,y} \boldsymbol{R}(x, y)$ if its similarity with the template is above a threshold, i.e., $\boldsymbol{R}(x^*, y^*) > s_{min}$.

This template matching scheme suffers practical problems in real applications: for instance, as image presents repetitive structures, the template can be matched in more than one point. Moreover, changes in point of view can affect the match result, due to changes in scale, orientation or warp of the patch. Although these limitations are important in a generic framework for feature tracking, considering a video sequence with small movements of the camera between frames (i.e., a sufficiently high frame rate) and non dramatic changes of the point of view, the method is suitable for a sufficiently reliable feature tracking system.

3.2.2.2 Local Planar Patches Warp

Let consider a slightly different setup of the feature tracking problem: suppose that patches are planar, i.e., the salient point in the image correspond to a point on a local plane in the scene. With reference to Figure 3.12, let consider two different cameras located at origins $\mathcal{A}$ and $\mathcal{B}$ with intrinsic projection matrix respectively $\mathbf{K}_A$ and $\mathbf{K}_B$ and relative position expressed by transformation $\mathbf{T}_{\mathcal{A}}^{\mathcal{B}}$. Lets name $\boldsymbol{I}_A$ and $\boldsymbol{I}_B$ respectively the images taken by the two cameras. Notice that this is a very generic setup: in a standard feature tracking problem the cameras are namely the same (i.e., $\mathbf{K}_A \equiv \mathbf{K}_B$), and the displacement is given by the motion in time.

Points of the patch lie on the plane π, identified with respect to the origin $\mathcal{A}$ by the homogeneous description of the plane $\pi^{(\mathcal{A})} = \begin{bmatrix}\overline{\mathbf{n}}^T & d\end{bmatrix}^T$, where $\overline{\mathbf{n}}$ is the unit vector normal to the plane and d is the distance from the origin $\mathcal{A}$ to the plane. It follows that a

3D cartesian point $\mathbf{P}^{(\mathcal{A})}$ lies on the plane $\pi^{(\mathcal{A})}$ when

$$\begin{bmatrix}\overline{\mathbf{n}}^T & d\end{bmatrix}\begin{bmatrix}\mathbf{P}^{(\mathcal{A})} \\ 1\end{bmatrix} = 0, \tag{3.38}$$

and it is projected to a homogeneous 2D point $\boldsymbol{p}^{(\mathcal{A})}$ in the image of camera $\mathcal{A}$ by

$$\boldsymbol{p}^{(\mathcal{A})} = \mathbf{K}_A \mathbf{P}^{(\mathcal{A})}. \tag{3.39}$$

Changing reference system, the point $\mathbf{P}^{(\mathcal{A})}$ is expressed with respect to the frame $\mathcal{B}$ as

$$\mathbf{P}^{(\mathcal{B})} = \mathbf{R}_{\mathcal{A}}^{\mathcal{B}} \mathbf{P}^{(\mathcal{A})} + \mathbf{t}_{\mathcal{A}}^{\mathcal{B}}, \tag{3.40}$$

and it is projected on $\boldsymbol{p}^{(\mathcal{B})}$ in the image of camera $\mathcal{B}$ by

$$\begin{aligned}\boldsymbol{p}^{(\mathcal{B})} &= \mathbf{K}_B \mathbf{P}^{(\mathcal{B})} &(3.41)\\ &= \mathbf{K}_B \mathbf{R}_{\mathcal{A}}^{\mathcal{B}} \mathbf{P}^{(\mathcal{A})} + \mathbf{K}_B \mathbf{t}_{\mathcal{A}}^{\mathcal{B}}. &(3.42)\end{aligned}$$

The mapping between $\boldsymbol{p}^{(\mathcal{A})}$ and $\boldsymbol{p}^{(\mathcal{B})}$, projection of a point that lies on the plane π, is given by an *homography*, i.e., a linear transformation expressed by a 3×3 matrix $\mathbf{H}_{\mathcal{A}}^{\mathcal{B}}$ such that

$$\boldsymbol{p}^{(\mathcal{B})} = \mathbf{H}_{\mathcal{A}}^{\mathcal{B}} \boldsymbol{p}^{(\mathcal{A})}. \tag{3.43}$$

Equation 3.38 can be rewritten in order to obtain $1 = -\frac{\overline{\mathbf{n}}^T \mathbf{P}^{(\mathcal{A})}}{d}$, thus Equation 3.42 can be rewritten as

$$\boldsymbol{p}^{(\mathcal{B})} = \mathbf{K}_B \left(\mathbf{R}_{\mathcal{A}}^{\mathcal{B}} - \frac{\mathbf{t}_{\mathcal{A}}^{\mathcal{B}} \overline{\mathbf{n}}^T}{d} \right) \mathbf{P}^{(\mathcal{A})}, \tag{3.44}$$

and substituted on the left member of Equation 3.43. The right member of Equation 3.43 is substituted with Equation 3.39, giving

$$\mathbf{H}_{\mathcal{A}}^{\mathcal{B}} = \mathbf{K}_B \left(\mathbf{R}_{\mathcal{A}}^{\mathcal{B}} - \frac{\mathbf{t}_{\mathcal{A}}^{\mathcal{B}} \overline{\mathbf{n}}^T}{d} \right) \mathbf{K}_A^{-1}, \tag{3.45}$$

from which it is possible to compute how the appearance of the patch around $\mathbf{P}^{(\mathcal{A})}$ changes (under the stated hypothesis). Consequently, it possible to warp the original image aiming at better feature matching results, which compensates for scale changes (e.g., when the camera walk up or away) and rotations.

The method for the patch warpint that we consider in this work, although it is not the unique possible choice is the following. Given a prediction of the feature location $\boldsymbol{p}^{(\mathcal{B})}$ on the second image and the size of the patch $w \times h$, we can identify the rectangular patch around the feature location as

$$\boldsymbol{p}_i^{(\mathcal{B})} = \boldsymbol{p}^{(\mathcal{B})} + \boldsymbol{p}_i,\ i = 1:4, \tag{3.46}$$

where

$$\boldsymbol{p}_1 = \begin{bmatrix} -\frac{w}{2} \\ -\frac{h}{2} \end{bmatrix} \qquad \boldsymbol{p}_2 = \begin{bmatrix} \frac{w}{2} \\ -\frac{h}{2} \end{bmatrix} \qquad \boldsymbol{p}_3 = \begin{bmatrix} \frac{w}{2} \\ \frac{h}{2} \end{bmatrix} \qquad \boldsymbol{p}_4 = \begin{bmatrix} -\frac{w}{2} \\ \frac{h}{2} \end{bmatrix}. \tag{3.47}$$

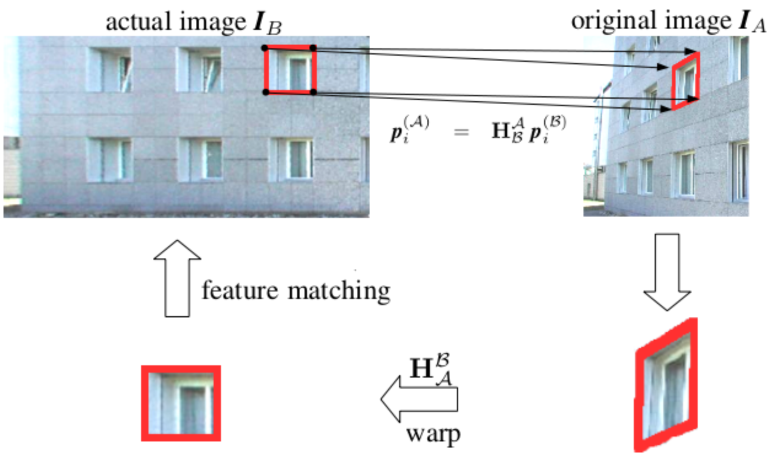

actual image I_B
original image I_A
$p_i^{(\mathcal{A})} = \mathbf{H}_{\mathcal{B}}^{\mathcal{A}} p_i^{(\mathcal{B})}$
feature matching
$\mathbf{H}_{\mathcal{A}}^{\mathcal{B}}$
warp

Figure 3.13: *Patch warping for matching process. A rectangular patch is extracted from the actual image $\mathbf{I}_B$ around the predicted position $\mathbf{p}^{(\mathcal{B})}$ of a feature, then the corresponding points ($\mathbf{p}_i^{(\mathcal{A})}$) in the original image $\mathbf{I}_A$ in which the feature was firstly perceived are computed through the homography $\mathbf{H}_{\mathcal{B}}^{\mathcal{A}}$. The patch extracted from the quadrilateral region described in the original image is warped with homography $\mathbf{H}_{\mathcal{A}}^{\mathcal{B}} = \mathbf{H}_{\mathcal{B}}^{\mathcal{A}\,-1}$ to a rectangular patch and then it can be used for the feature matching in the actual image.*

We can compute the location of these point in the first image, i.e., in the image in which the feature was firstly perceived by the camera $\mathcal{A}$, as

$$p_i^{(\mathcal{A})} = \mathbf{H}_{\mathcal{B}}^{\mathcal{A}} p_i^{(\mathcal{B})}, \tag{3.48}$$

where $\mathbf{H}_{\mathcal{B}}^{\mathcal{A}} = \mathbf{H}_{\mathcal{A}}^{\mathcal{B}\,-1}$. The four points $p_i^{(\mathcal{A})}$ determines a quadrilateral area Q in the image I_A that represents the template that has to be matched in the new image after being warped to a rectangular patch of $w \times h$ pixels. This process is schematized in Figure 3.13. With simple changes, this procedure can be used to compensate distortion effects too: Equation 3.43 can be modified in order to consider the non linear transformation induced by the projection and the distortion model (i.e., the functions $K_A(\cdot)$ and $K_B(\cdot)^{-1}$ introduced in Section 3.1.7). The warp of the patch has to be computed as a mapping between all points contained in the source Q area to the destination rectangular area. Notice that when a new feature is perceived we need to store a patch around it which has to be large enough for subsequent warping.

An obscure point of this procedure, at least with the concepts reported up to here, is how it is possible to know the relative camera position $\mathbf{T}_{\mathcal{A}}^{\mathcal{B}}$, the point $\mathbf{P}^{(\mathcal{A})}$ location and the local plane of the landmark $\pi^{(\mathcal{A})}$. These things will be explained later (see Chapters5 and 6); for now, it is sufficient to say that $\mathbf{T}_{\mathcal{A}}^{\mathcal{B}}$ and $\mathbf{P}^{(\mathcal{A})}$ will be adequately estimated, while $\pi^{(\mathcal{A})}$ is approximated in the following way. Considering that feature was firstly perceived in the image of camera $\mathcal{A}$ on point $p^{(\mathcal{A})}$, thanks to Equation 3.18 the direction of the interpretation line $\overline{\mathbf{d}}_A^{(\mathcal{A})}$ can be computed. Similarly, given the predicted position of the feature $p^{(\mathcal{B})}$ in the second image, $\overline{\mathbf{d}}_B^{(\mathcal{B})}$ can be computed and expressed in the $\mathcal{A}$ reference frame $\overline{\mathbf{d}}_B^{(\mathcal{A})}$ through Equation 2.16. The normal to the plane of the feature is approximated, alike in [15] by the mean of the opposite of the two considered direction vector:

$$\overline{\mathbf{n}} \;=\; -\frac{1}{2}\left(\overline{\mathbf{d}}_A^{(\mathcal{A})} + \overline{\mathbf{d}}_B^{(\mathcal{B})}\right). \tag{3.49}$$

The geometrical interpretation of this choice is to consider the plane oriented as the bisector of the angles formed by the two viewing ray. The last element that has to be specified is the distance d of the plain, that could be recovered by Equation 3.38.

Obviously this simple method suffers from limitations and approximation. First of all, it is quite far from the reality that regions around corners are planar: most of them are located on vertex of 3D objects, thus directions of edges have great differences. Secondly, though the local plane is a good approximation, its normal is obviously not properly estimated by the proposed approximation.

CHAPTER 4

Filtering and Optimization

This chapter introduces, after a brief recall on *probability theory*, two basic techniques (Extended Kalman Filter and Non Linear Least Squares Optimization) that are largely studied and applied in the solution of SLAM problems. Here only a general introduction to these techniques is provided, while specific adaptations to the SLAM problem are postponed to the next chapter.

4.1 Brief Probability Recall

Probability theory is the branch of mathematics concerned with the analysis of random phenomena. Probability is a measure of the expectation that an event will occur. The higher the probability of an event, the more certain we are that the event will occur. The next sections are a brief summary of basics concepts of probability theory, for a more detailed description refer to [3] and [72].

4.1.1 Axioms of Probability

Considering a process with a random outcome (let us call it an *experiment*), an event A is a set of the possible outcomes of the experiment itself. For instance, consider the outcome of a rolling die. The possible events are “the outcome is 4”, “the outcome is an odd number” and so on. The probability $\boldsymbol{p}(A)$ must satisfy the following axioms of probability:

- It is nonnegative: $\boldsymbol{p}(A) \geq 0 \quad \forall A$.

- It is equal to 1 if $A = S$, where S is the certain event, i.e., the event which contains all the possible outcomes of the considered experiment: $\boldsymbol{p}(S) = 1$.
- Considering two mutually exclusive events A and B (i.e. A and B do not have any common element $A \cap B = \emptyset$), the probability of their union is the sum of the event probabilities:

$$\boldsymbol{p}(A \cup B) \quad = \quad \boldsymbol{p}(A) + \boldsymbol{p}(B). \tag{4.1}$$

- It follows that, considering the complementary event of A: $\overline{A} = S \backslash A$, where $\backslash$ is the set subtraction operation,

$$\boldsymbol{p}(\overline{A}) \quad = \quad 1 - \boldsymbol{p}(A), \tag{4.2}$$

and since $\emptyset \equiv \overline{S}$,

$$\boldsymbol{p}(\emptyset) \quad = \quad 0, \tag{4.3}$$

thus, $\emptyset$ is the impossible event.

4.1.2 Random Variables and Probability Density Function

A *random variable* can be tough as a mathematical variable with unknown or uncertain values. Uncertainty can be due to the fact that an event is not yet occurred and it can not be predicted with certainty (e.g., the weather of tomorrow) or the imperfect knowledge of the real value (e.g., the measurement coming from an instrument subject to noises and errors). A *continuous* random variable x assume a real value $x = \varepsilon \in \mathbb{R}$ and it has an associated *probability density function (pdf)* (or *density*) $p_x(\varepsilon)$ defined as

$$p_x(\varepsilon) \quad = \quad \lim_{d\varepsilon \to 0} \frac{\boldsymbol{p}(\varepsilon - d\varepsilon < x \leq \varepsilon)}{d\varepsilon}. \tag{4.4}$$

It follows that the probability that x assumes values in a range $(a, b]$ is

$$\boldsymbol{p}(a < x \leq b) \quad = \quad \int_a^b p_x(x)dx. \tag{4.5}$$

The *cumulative distribution function (cdf)* is defined as

$$F_x(b) \quad = \quad \boldsymbol{p}(x \leq b) = \int_{-\infty}^b p_x(x)dx, \tag{4.6}$$

and its limit to the infinity is 1, since the certain event S is $x \leq \infty$:

$$\lim_{b \to \infty} F_x(b) \quad = \quad \boldsymbol{p}(-\infty < x \leq \infty) = 1. \tag{4.7}$$

The *expected value* of a random variable, also known as its *mean* or *first moment* is

$$\mu \quad = \quad \mathrm{E}[x] \quad = \quad \int_{-\infty}^{+\infty} x \, p_x(x)dx, \tag{4.8}$$

and the *variance* is defined as

$$\sigma^2 \; = \; \mathrm{VAR}(x) \; = \; \mathrm{E}[(x - \mathrm{E}[x])^2] \; = \; \int_{-\infty}^{+\infty} (x - \mathrm{E}[x])^2 \, p_x(x)dx, \tag{4.9}$$

where $\sigma = \sqrt{\sigma^2}$ is called *standard deviation*.

4.1.3 Joint Probability

Considering two random variables x and y, it is possible to define their *joint probability density* similarly to Equation 4.4

$$p_{xy}(\varepsilon,\eta) = \lim_{d\varepsilon\to 0, d\eta\to 0} \frac{\boldsymbol{p}(\{\varepsilon - d\varepsilon < x \le \varepsilon\} \cap \{\eta - d\eta < y \le \eta\})}{d\varepsilon\, d\eta}. \quad (4.10)$$

Marginalization recovers the individual pdf (called *marginal pdf*) by integrating the joint pdf over one of the two variables:

$$p_x(\varepsilon) = \int_{-\infty}^{+\infty} p_{xy}(\varepsilon, y)dy, \quad (4.11)$$

$$p_y(\eta) = \int_{-\infty}^{+\infty} p_{xy}(x, \eta)dx. \quad (4.12)$$

Similarly to Equation 4.6, the joint cdf is defined as

$$F_{xy}(a,b) = \boldsymbol{p}(x \le a,\ y \le b) = \int_{-\infty}^{a}\int_{-\infty}^{b} p_{xy}(x,y)dx\,dy, \quad (4.13)$$

and its limit to the infinity is 1:

$$\lim_{x\to+\infty, y\to+\infty} F_{xy}(x,y) = 1. \quad (4.14)$$

Besides individual mean values and variances $\mu_x = \mathrm{E}[x]$, $\mu_y = \mathrm{E}[y]$, $\sigma_x^2 = VAR[x]$ and $\sigma_y^2 = VAR[y]$, that can be computed after the marginalization, it is possible to define the *covariance* as

$$\sigma_{xy} = \mathrm{COV}(x,y) = \mathrm{E}[(x - \mathrm{E}[x])(y - \mathrm{E}[y])] \quad (4.15)$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{+\infty} (x - \mathrm{E}[x])(y - \mathrm{E}[y])\, p_{xy}(x,y)\, dx\, dy, \quad (4.16)$$

for which it holds that $\mathrm{COV}(x,y) = \mathrm{COV}(y,x)$.

The *correlation* index, defined as

$$\rho_{xy} = \frac{\mathrm{COV}(x,y)}{\sqrt{\mathrm{VAR}(x)\,\mathrm{VAR}(y)}}, \quad (4.17)$$

measures the linear relation existing between the two variables. It lies in the interval $[-1,1]$ and if its value is ± 1 the two variables are linearly dependent ($y = ax + b$ with $a \neq 0$), which implies that the value of one of the two variables is sufficient to know the value of the other one. If the correlation between two random variables is zero, i.e., the covariance is zero, the variables are *independent*, i.e. the knowledge on the value of one of the two variable gives no information on the possible outcome of the second. Moreover, the expected value of the product of two independent variables is equal to the product of their expected values if and only if they are independent:

$$\mathrm{E}[x\,y] = \mathrm{E}[x]\,\mathrm{E}[y] \iff x, y \quad \text{are independent.} \quad (4.18)$$

The joint density of two independent random variables is the product of their marginals and consequently, considering n independent random variables, the product of their marginals is the joint density:

$$p_{xy}(x,y) = p_x(x)p_y(y) \Longleftrightarrow x,y \quad \text{are independent,} \tag{4.19}$$

$$p_{x_1,\ldots,x_n}(x_1,\ldots,x_n) = \prod_{i=1}^{n} p_{x_i}(x_i) \Longleftrightarrow x_1,\ldots x_n \quad \text{are independent.} \tag{4.20}$$

4.1.4 Random Vectors

A natural extension of the joint probability of n variables is to consider variables organized in a *random vector*

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \tag{4.21}$$

and to express its joint probability as

$$p_{\mathbf{x}}(\mathbf{x}) = p_{x_1,\ldots,x_n}(x_1,\ldots,x_n). \tag{4.22}$$

The mean of a random vector is defined as the vector of individual means

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{x}] = [\mathrm{E}[x_1], \mathrm{E}[x_2], \ldots, \mathrm{E}[x_n]]^T, \tag{4.23}$$

while its covariance is a matrix computed as $\mathrm{COV}[\mathbf{x}] = \mathrm{E}[(\mathbf{x} - \mathrm{E}[(\mathbf{x})])((\mathbf{x} - \mathrm{E}[(\mathbf{x})]))^T]$:

$$\mathrm{COV}[\mathbf{x}] = \begin{bmatrix} \mathrm{VAR}[x_1] & \mathrm{COV}[x_1,x_2] & \ldots & \mathrm{COV}[x_1,x_n] \\ \mathrm{COV}[x_2,x_1] & \mathrm{VAR}[x_2] & \ldots & \mathrm{COV}[x_2,x_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{COV}[x_n,x_1] & \mathrm{COV}[x_n,x_2] & \ldots & \mathrm{VAR}[x_n,x_n] \end{bmatrix}, \tag{4.24}$$

The covariance matrix is usually referred with the $\boldsymbol{\Sigma}$ symbol, while elements are referred with the lowercase σ_{ij}, indicating in subscript the matrix element, except for the diagonal elements, which corresponds to the individual variable variance, indicated with σ_i^2 (alternatively they can be identified with σ_{ii}):

$$\boldsymbol{\Sigma} = \mathrm{COV}[\mathbf{x}] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{12} & \sigma_2^2 & \ldots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1n} & \sigma_{2n} & \ldots & \sigma_n^2 \end{bmatrix}. \tag{4.25}$$

The covariance matrix is symmetric. If there are no linear dependences between variables the covariance matrix is positive definite, otherwise it is semidefinite positive.

4.1.5 Conditional Probability and Bayes Formula

The *conditional probability* of an event A given an event B is defined as

$$\boldsymbol{p}(A|B) = \frac{\boldsymbol{p}(A,B)}{\boldsymbol{p}(B)}, \tag{4.26}$$

and similarly, the conditioned density of a variable x given y is

$$p_{x|y}(x,y) = \frac{p_{xy}(x,y)}{p_y(y)}. \tag{4.27}$$

It is noticeable that if two variables are independent, the Equation 4.27 and Equation 4.19 give $p_{x|y}(x,y) = p_x(x)$, since the knowledge of y gives no additional informations on x.

With the aims of a simplification of the notation, the precise writing of the distribution function are usually discarded and the notation is simplified as in the following:

$$p_x(x) \equiv \boldsymbol{p}(x) \tag{4.28}$$
$$p_{xy}(x,y) \equiv \boldsymbol{p}(x,y) \tag{4.29}$$
$$p_{x|y}(x,y) \equiv \boldsymbol{p}(x|y) \tag{4.30}$$

i.e., the arguments of the density function are discarded and the meaning of the density is put in the arguments. This notation allows to treat probability of events and random variables in a unified manner.

The *total probability theorem* states that marginal distribution of random variables can be obtained by conditional probability as

$$\boldsymbol{p}(x) = \int_{-\infty}^{+\infty} \boldsymbol{p}(x,y)dy = \int_{-\infty}^{+\infty} \boldsymbol{p}(x|y)\boldsymbol{p}(y)dy, \tag{4.31}$$

thanks to the definition of conditioned density of Equation 4.27.

The conditional density of variable x given y can be expressed by its reverse conditioning through the *Bayes Formula*

$$\boldsymbol{p}(x|y) = \frac{\boldsymbol{p}(y|x)\boldsymbol{p}(x)}{\boldsymbol{p}(y)}. \tag{4.32}$$

A common interpretation of this formula calls $\boldsymbol{p}(x|y)$ the *posterior density function* and $\boldsymbol{p}(x)$ the *prior*:

- the prior $\boldsymbol{p}(x)$ represent an *initial degree of belief* on the value of the random variable x,
- $\boldsymbol{p}(y|x)$ is the *evidence from the data*, i.e., the likelihood of observing y given x,
- the combination of prior belief with the evidence from the data gives the *posterior* $\boldsymbol{p}(x|y)$.

4.1.6 Conditional Independence

Consider the more general case of the conditioned density $\boldsymbol{p}(x|y,z)$, i.e, the probability density function of a variable x given the couple of variables y and z. The generalization of Equation 4.27 gives

$$\boldsymbol{p}(x|y,z) = \frac{\boldsymbol{p}(x,y|z)}{\boldsymbol{p}(x|y)}. \tag{4.33}$$

and x and y are said to be *conditionally independent* given z if

$$\boldsymbol{p}(x,y|z) = \boldsymbol{p}(x|z)\boldsymbol{p}(y|z), \tag{4.34}$$

or equivalently

$$\boldsymbol{p}(x|y,z) = \boldsymbol{p}(x|z), \tag{4.35}$$

i.e., y does not provide any additional information on x if z is given. The conditional independence does not imply the independence, i.e., in general $\boldsymbol{p}(xy) \neq \boldsymbol{p}(x)\boldsymbol{p}(y)$.

4.1.7 Gaussian Random Variables and Vectors

A variable x is a *Gaussian* or *Normal* random variable if its pdf has the form

$$p_x(x) = \mathcal{N}(\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}} \tag{4.36}$$

where the arguments are the *mean* μ and the standard deviation σ. The notation $x \sim \mathcal{N}(\mu,\sigma)$ is used to indicate that x has a Gaussian pdf. Its generalization to random vectors is indicated with $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ where n is the number of variables in the vector $\mathbf{x}$, $\boldsymbol{\mu}$ is the n element vector of means and $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix. The joint pdf is

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}), \tag{4.37}$$

$$p_{\mathbf{x}}(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}\ \det(\boldsymbol{\Sigma})^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{4.38}$$

A very important properties of Gaussian random variables is that they remains Gaussian under linear transformations. Let consider $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, i.e., a Gaussian random vector with n elements, $\mathbf{A}$ an $m \times n$ matrix and $\mathbf{b}$ an n elements real vector. The vector defined by

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{4.39}$$

is a Gaussian random vector with the following parameters:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T). \tag{4.40}$$

Moreover, Gaussian Random Vectors have nice properties considering their conditional probability. Let consider a Gaussian random vector $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}},\boldsymbol{\Sigma}_{\mathbf{y}})$ partitioned in two vectors $\mathbf{x}$ and $\mathbf{z}$:

$$\mathbf{y} = \begin{bmatrix}\mathbf{x}\\ \mathbf{z}\end{bmatrix}. \tag{4.41}$$

It is easy to notice that the mean vector can be easily decomposed in

$$\boldsymbol{\mu}_{\mathbf{y}} = \mathrm{E}[\mathbf{y}] = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{x}} \\ \boldsymbol{\mu}_{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathrm{E}[\mathrm{x}] \\ \mathrm{E}[\mathrm{z}] \end{bmatrix}, \tag{4.42}$$

and the covariance matrix can be partitioned and expressed as a block matrix

$$\boldsymbol{\Sigma}_{\mathbf{yy}} = \mathrm{COV}(\mathbf{y}) = \begin{bmatrix} \mathrm{COV}(\mathbf{x},\mathbf{x}) & \mathrm{COV}(\mathbf{x},\mathbf{z}) \\ \mathrm{COV}(\mathbf{z},\mathbf{x}) & \mathrm{COV}(\mathbf{z},\mathbf{z}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{xx}} & \boldsymbol{\Sigma}_{\mathbf{xz}} \\ \boldsymbol{\Sigma}_{\mathbf{xz}}^T & \boldsymbol{\Sigma}_{\mathbf{zz}} \end{bmatrix}, \tag{4.43}$$

where $\mathrm{COV}(\mathbf{z},\mathbf{x}) = \mathrm{COV}(\mathbf{x},\mathbf{z})^T$ due to the symmetries of the covariance matrix.

The conditional pdf of $\mathbf{x}$ given $\mathbf{z}$ when $\mathbf{x}$ and $\mathbf{z}$ are jointly Gaussian is

$$p_{\mathbf{x}|\mathbf{z}}(\mathbf{x},\mathbf{z}) = \frac{p_{\mathbf{xz}}(\mathbf{x},\mathbf{z})}{p_{\mathbf{z}}(\mathbf{z})} = \frac{\frac{1}{\sqrt{2\pi\ \det(\boldsymbol{\Sigma}_{\mathbf{yy}})}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{y}}\right)^T \boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{y}}\right)\right\}}{\frac{1}{\sqrt{2\pi\ \det(\boldsymbol{\Sigma}_{\mathbf{zz}})}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{z}}\right)^T \boldsymbol{\Sigma}_{\mathbf{zz}}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{z}}\right)\right\}}, \tag{4.44}$$

and it is possible to demonstrate, after some manipulation, that it remains a Gaussian pdf with the following parameters

$$p_{\mathbf{x}|\mathbf{z}}(\mathbf{x},\mathbf{z}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{z}}), \tag{4.45}$$

$$\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}} = \boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xz}}\boldsymbol{\Sigma}_{\mathbf{zz}}^{-1}(\mathbf{z}-\boldsymbol{\mu}_{\mathbf{z}}), \tag{4.46}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{z}} = \boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xz}}\boldsymbol{\Sigma}_{\mathbf{zz}}^{-1}\boldsymbol{\Sigma}_{\mathbf{zx}}. \tag{4.47}$$

The above relations are usually called *fundamental equation of linear estimation.*

4.1.8 Chi-Square Distributed Random Variables

Given a n-dimensional Gaussian random vector $\mathbf{x}$ with mean $\mathrm{E}[\mathbf{x}] = \boldsymbol{\mu}_x$ and covariance matrix $\mathrm{COV}[\mathbf{x}] = \boldsymbol{\Sigma}_x$, the Gaussian (scalar) variable that results from the quadratic form

$$y = (\mathbf{x}-\boldsymbol{\mu}_x)^T \boldsymbol{\Sigma}_x^{-1}(\mathbf{x}-\boldsymbol{\mu}_x), \tag{4.48}$$

can be shown to be the sum of the squares of n independent zero-mean unit-variance Gaussian random variables, resulting in a chi-square distribution with n degrees of freedom. The chi-square distribution is usually indicated as

$$y \sim \chi_n^2. \tag{4.49}$$

The mean and covariance of y are respectively $\mathrm{E}[y] = n$ and $\mathrm{VAR}[y] = 2n$. The cumulative distribution function of the chi-square distribution is expressed with the function $F_{\chi_n^2}(q)$, thus the probability of $y < q$ can be expressed as

$$\boldsymbol{p}(-\infty < y \leq q) = F_{\chi_n^2}(q). \tag{4.50}$$

The inverse of the cdf, expressed with $F_{\chi_n^2}^{-1}(\alpha)$, expresses, given a probability $\alpha \in [0,1]$, the value which the random variable will be at, or below, with α probability.

4.1.8.1 Mahalanobis Distance

The Mahalanobis distance, introduced in [58] by P. C. Mahalanobis, measures the distance between a sampled random vector and its expected value taking into accounts the covariances between its elements. Considering a random vector $\mathbf{x}$ with mean vector $\boldsymbol{\mu}_x$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}}$, the Mahalanobis distance of $\mathbf{x}$ from $\boldsymbol{\mu}_x$ is defined as

$$d = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_x)^T \boldsymbol{\Sigma}_x^{-1} (\mathbf{x} - \boldsymbol{\mu}_x)}. \tag{4.51}$$

It can be noticed that, if the $\mathbf{x}$ random vector is distributed as a multivariate Gaussian random vector with mean $\boldsymbol{\mu}_x$ and covariance matrix $\boldsymbol{\Sigma}_x$, the square value of the Mahalanobis distance (d^2) is distributed as a chi-square variable with n degrees of freedom, where n is the cardinality of $\mathbf{x}$ vector:

$$d^2 \sim \chi_n^2. \tag{4.52}$$

The Mahalanobis distance is equivalent to the Euclidean distance if the covariance matrix is the identity matrix, i.e., if $\mathbf{x}$ is composed by n independent random variables with variance equal to 1. When the covariance matrix is diagonal, i.e., variables of the random vector are independent, but with different variance, the Mahalanobis distance results in an Euclidean distance weighted with the standard deviation of each variable. In the general cases, the Mahalanobis distance allows to evaluate the distance in terms of probability: two samples $\mathbf{x}_1$ and $\mathbf{x}_2$ of the random vector $\mathbf{x}$ have the same Mahalanobis distance if they are equally distant from the mean value $\boldsymbol{\mu}_x$ with the same probability density. This concept is intuitively described in Figure 4.1, where the Mahalanobis distance is compared with the Euclidean distance in a bivariate case.

4.2 Bayesian Filtering and the Extended Kalman Filter

The *Bayes Filter* (or *Recursive Bayesian estimator*) is a general probabilistic approach for estimating an unknown probabilistic density function using indirect measurements and a model of state transition. A particular implementation of the Bayesian filter is the Kalman Filter, that works with linear models acting on continuous variables with Gaussian distribution. The Extended Kalman Filter works on nonlinear models by applying linearizations. The materials presented in this section can be deepened in [102], [3] and [34].

4.2.1 General Bayes Filter

Consider a generic multivariate random variable $\mathbf{x}_k$ that changes over the discrete time k and measurements $\mathbf{z}_k$ that are observation of the state itself. Under the hypothesis that the state evolution is a Markov process, the probability density function of the state at current time given the previous state value is conditionally independent on the earlier states, measurements and inputs:

$$\boldsymbol{p}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \ldots, \mathbf{x}_0, \mathbf{z}_{k-1}, \ldots, \mathbf{z}_1,) \quad = \tag{4.53}$$

$$\boldsymbol{p}(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{z}_{1:k-1}) \quad = \quad \boldsymbol{p}(\mathbf{x}_k | \mathbf{x}_{k-1}), \tag{4.54}$$

where $\mathbf{x}_{0:k-1}$ is a compact notation for $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-2}, \mathbf{x}_{k-1}$. This hypothesis is also known as *state completeness*, since the knowledge of the last state allows to predict the

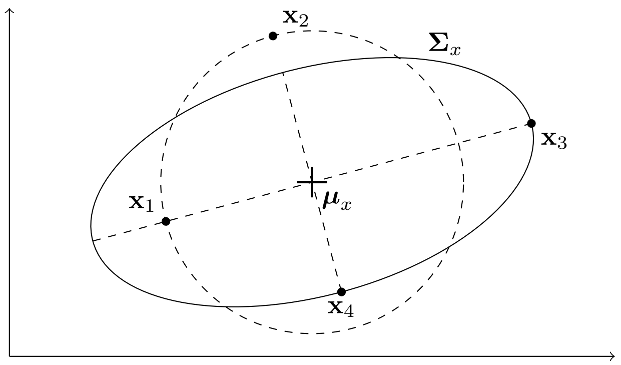


Figure 4.1: *Comparison of Euclidean and Mahalanobis distances. The bivariate random vector* $\mathbf{x}$ *with the* 2×2 *covariance matrix* $\mathbf{\Sigma}_x$ *and the mean vector* $\boldsymbol{\mu}_x$ *is represented by the ellipse centered in* $\boldsymbol{\mu}_x$ *that represents the contours of the density function. Let consider the pair of sampled points* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*. They lie on the same circumference centered in* $\boldsymbol{\mu}_x$*, thus they are equidistant from the mean* $\boldsymbol{\mu}_x$ *in terms of Euclidean distance but* $\mathbf{x}_1$ *is closest to the mean than* $\mathbf{x}_2$ *in terms of Mahalanobis distance, i.e., it is more probable to obtain the sample* $\mathbf{x}_1$ *than* $\mathbf{x_2}$ *from the random vector* $\mathbf{x}$*. Consider now the points* $\mathbf{x}_3$ *and* $\mathbf{x}_4$*. Theirs Euclidean distance from the mean is obviously different, while the Mahalanobis distance is the same, since they lies on the ellipse described by* $\mathbf{\Sigma}_x$ *centered in the mean. Consequently it is equally probable to obtain the samples* $\mathbf{x}_3$ *and* $\mathbf{x}_4$*.*

current state in the best way. $\boldsymbol{p}(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *state transition probability* and it describes the stochastic evolution of the state.

Furthermore, if measurements are observed states of an Hidden Markov Model (HMM), i.e., measurements at current time are conditionally independent of all other states given the current state, the state $\mathbf{x}_k$ is sufficient to predict in the best way the measure $\mathbf{z}_k$ through *measurement probability*:

$$\boldsymbol{p}(\mathbf{z}_k|\mathbf{x}_{0:k}, \mathbf{z}_{1:k-1}) \quad = \quad \boldsymbol{p}(\mathbf{z}_k|\mathbf{x}_k). \tag{4.55}$$

The Bayesian optimal filter computes the *posterior distribution*,

$$\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k}), \tag{4.56}$$

i.e., the distribution of probability of the state given all the measurement up to the current time given the prior distribution $\boldsymbol{p}(\mathbf{x}_0)$ and the complete measurement sequence $\mathbf{z}_{1:k}$ through two steps: the *prediction step* and the *update step*.

The *prediction step* aims to calculate the probability distribution $\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})$, i.e., the distribution probability of $\mathbf{x}_k$ using past measurements (up to $\mathbf{z}_{1:k-1}$). Assuming to know the distribution

$$\boldsymbol{p}(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}), \tag{4.57}$$

i.e., the posterior distribution at the previous time step, the joint distribution of the current and the previous state can be computed as

$$\begin{aligned} \boldsymbol{p}(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) &= \boldsymbol{p}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k-1})\boldsymbol{p}(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) && (4.58)\\ &= \boldsymbol{p}(\mathbf{x}_k|\mathbf{x}_{k-1})\boldsymbol{p}(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}), && (4.59)\end{aligned}$$

and integration over all possible $\mathbf{x}_{k-1}$ gives

$$\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int \boldsymbol{p}(\mathbf{x}_k|\mathbf{x}_{k-1})\boldsymbol{p}(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1}, \tag{4.60}$$

that is desired distribution of the prediction step.

The *update step* complete the calculation of the posterior distribution $\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k})$, that can be expressed, thanks to the *Bayes Rule*, in the recursive form

$$\begin{aligned} \boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k}) &= \frac{\boldsymbol{p}(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{1:k-1})\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{\boldsymbol{p}(\mathbf{z}_k|\mathbf{z}_{1:k-1})} && (4.61)\\ &= \frac{\boldsymbol{p}(\mathbf{z}_k|\mathbf{x}_k)\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{\int \boldsymbol{p}(\mathbf{z}_k|\mathbf{x}_k)\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})d\mathbf{x}_k}. && (4.62)\end{aligned}$$

The update step aims to integrate the current measure $\mathbf{z}_k$ to refine the prediction carried out at the first step.

4.2.2 Kalman Filter

The *Kalman Filter* and generally the family of *Gaussian Filters* constitute the earliest tractable implementation of the Bayes Filter for continuous state space. It was introduced by R. E. Kálmán in 1960 in [44] and it has been the subject of extensive research and applications, particularly in the area of autonomous or assisted navigation. The underlying basic hypothesis is that the posterior distribution of Equation 4.56 is a multivariate normal distribution. This hypothesis is verified under the recurrent formula of Equation 4.62 if the three next assumptions are valid:

- The initial probability distribution follows a Gaussian distribution

  $$\boldsymbol{p}(\mathbf{x}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \tag{4.63}$$

  with $\boldsymbol{\mu}_0$ the n dimensional vector of the initial mean and $\boldsymbol{\Sigma}_0$ the $n \times n$ matrix of the initial covariance.

- The *state transition probability* is a linear function of Gaussian variables

  $$\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \boldsymbol{\eta}_k, \tag{4.64}$$

  where

  - $\mathbf{A}_k$ is a $n \times n$ time dependent matrix that relates the previous state with the current state,
  - $\mathbf{x}_{k-1}$ is the previous state, normally distributed by assumption,
  - $\mathbf{u}_k$ is a p vector of deterministic external inputs,

  - $\mathbf{B}_k$ is a $n \times p$ matrix that express how current inputs affects the current state,
  - $\boldsymbol{\eta}_k$ is a zero-mean n elements Gaussian variable that represent unmodeled noises, i.e., $\boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\eta}_k}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}_k})$

- The *measurement probability* is a linear function of Gaussian variables

$$\mathbf{z}_k = \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\delta}_k, \tag{4.65}$$

  where

  - $\mathbf{C}_k$ is a $m \times n$ matrix that transforms the state to the measurement
  - $\boldsymbol{\delta}_k$ is a zero-mean m elements Gaussian variable that represents unmodeled noises in the measurement process, i.e., $\boldsymbol{\delta}_k \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\delta}_k}, \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k})$, and it is uncorrelated with $\boldsymbol{\eta}_k$.

These assumptions make easy the computation of the distribution after the *prediction step* and after the *update step*. The prediction step stated in Equation 4.64 is a linear function of Gaussian variables and deterministic terms, thus the distribution $\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ can be easily computed as the resulting Gaussian variable

$$\begin{aligned} \boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1}) &\sim \mathcal{N}(\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k), & (4.66)\\ \hat{\boldsymbol{\mu}}_k &= \mathbf{A}_k \boldsymbol{\mu}_{k-1} + \mathbf{B}_k \mathbf{u}_k, & (4.67)\\ \hat{\boldsymbol{\Sigma}}_k &= \mathbf{A}_k \boldsymbol{\Sigma}_{k-1} \mathbf{A}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}. & (4.68) \end{aligned}$$

For the calculation of the update step, let consider a new variable $\boldsymbol{\epsilon} = [\mathbf{x}_k^T,\ \mathbf{z}_k^T]^T$. It is a $n+m$ elements multivariate Gaussian variable $\boldsymbol{p}(\boldsymbol{\epsilon}) = \boldsymbol{p}(\mathbf{x}_k, \mathbf{z}_k|\mathbf{z}_{1:k-1}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\epsilon}}, \Sigma_{\boldsymbol{\epsilon}})$ with

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\epsilon}} &= \begin{bmatrix} \hat{\boldsymbol{\mu}}_k \\ \mathbf{C}_k \hat{\boldsymbol{\mu}}_k \end{bmatrix}, & (4.69)\\ \Sigma_{\boldsymbol{\epsilon}} &= \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_k & \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T \\ \mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k & \mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k} \end{bmatrix}. & (4.70) \end{aligned}$$

To compute the result after the update step $\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k})$ it is possible to condition $\mathbf{x}_k$ with $\mathbf{z}_k$, that is a new Gaussian variable with

$$\begin{aligned} \boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_k) &\sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), & (4.71)\\ \boldsymbol{\mu}_k &= \hat{\boldsymbol{\mu}}_k + \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T \left(\mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}\right)^{-1} (\mathbf{z}_k - \mathbf{C}_k \hat{\boldsymbol{\mu}}_k), & (4.72)\\ \boldsymbol{\Sigma}_k &= \hat{\boldsymbol{\Sigma}}_k - \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T \left(\mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}\right)^{-1} \mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k, & (4.73) \end{aligned}$$

that could be rewritten as

$$\begin{aligned} \mathbf{S} &= \mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}, & (4.74)\\ \mathbf{K} &= \hat{\boldsymbol{\Sigma}}_k \mathbf{C}_k^T \mathbf{S}^{-1}, & (4.75)\\ \boldsymbol{\mu}_k &= \hat{\boldsymbol{\mu}}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{C}_k \hat{\boldsymbol{\mu}}_k), & (4.76)\\ \boldsymbol{\Sigma}_k &= \hat{\boldsymbol{\Sigma}}_k - \mathbf{K}\mathbf{C}_k \hat{\boldsymbol{\Sigma}}_k, & (4.77) \end{aligned}$$

where $\mathbf{S}$ is the covariance matrix of the Gaussian distribution $\boldsymbol{p}(\mathbf{z}_k|\mathbf{z}_{1:k-1})$ and $\mathbf{K}$ is the *Kalman gain*, which aims to correct the mean of the estimation thanks to the difference between the effective values of $\mathbf{z}_k$ and its prediction from state ($\mathbf{C}_k\hat{\boldsymbol{\mu}}_k$) and correct the covariance estimation too.

The complete algorithm that performs an estimation step in the Kalman Filter is sketched in Algorithm 2.

Algorithm 2 $[\boldsymbol{\mu}_k, \Sigma_k]$ = KF($\boldsymbol{\mu}_{k-1}$, $\boldsymbol{\Sigma}_{k-1}$, $\mathbf{u}_k$, $\mathbf{z}_k$, $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}$, $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}$)

```
1: $\hat{\boldsymbol{\mu}}_k = \mathbf{A}_k\boldsymbol{\mu}_{k-1} + \mathbf{B}_k\mathbf{u}_k$
2: $\hat{\boldsymbol{\Sigma}}_k = \mathbf{A}_k\boldsymbol{\Sigma}_{k-1}\mathbf{A}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}$
3: $\mathbf{S} = \mathbf{C}_k\hat{\boldsymbol{\Sigma}}_k\mathbf{C}_k^T + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}$
4: $\mathbf{K} = \hat{\boldsymbol{\Sigma}}_k\mathbf{C}_k^T\mathbf{S}^{-1}$
5: $\boldsymbol{\mu}_k = \hat{\boldsymbol{\mu}}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{C}_k\hat{\boldsymbol{\mu}}_k)$
6: $\boldsymbol{\Sigma}_k = \hat{\boldsymbol{\Sigma}}_k - \mathbf{K}\mathbf{C}_k\hat{\boldsymbol{\Sigma}}_k$
```

4.2.3 Extended Kalman Filter

Most real systems do not comply with the assumptions made for the Kalman filter. Especially, the linear relation in the state transition function and in the measurement process is too strict for most of real problems. Thus, a simple solution that allows to treat nonlinear functions is to linearize them through first order Taylor expansion.

Let consider the state transition governed by a generic function $f(\cdot)$ such that

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k). \tag{4.78}$$

The prediction step can be performed as

$$\boldsymbol{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k), \tag{4.79}$$
$$\hat{\boldsymbol{\mu}}_k = f(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k = \mathbf{0}) \tag{4.80}$$
$$\hat{\boldsymbol{\Sigma}}_k \simeq \mathbf{F}_{\mathbf{x}}\boldsymbol{\Sigma}_{k-1}\mathbf{F}_{\mathbf{x}}^T + \mathbf{F}_{\boldsymbol{\eta}}\boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}\mathbf{F}_{\boldsymbol{\eta}}^T \tag{4.81}$$

where

$$\mathbf{F}_{\mathbf{x}} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \boldsymbol{\eta})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}_{k-1}, \mathbf{u}=\mathbf{u}_k, \boldsymbol{\eta}=\mathbf{0}}, \tag{4.82}$$

is the Jacobian of the function $f(\cdot)$ with respect to the state variable evaluated at the last estimation of the state mean $\boldsymbol{\mu}_{k-1}$, with the current input $\mathbf{u}_k$ and assuming no noises;

$$\mathbf{F}_{\boldsymbol{\eta}} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\right|_{\mathbf{x}=\boldsymbol{\mu}_{k-1}, \mathbf{u}=\mathbf{u}_k, \boldsymbol{\eta}=\mathbf{0}}, \tag{4.83}$$

is the Jacobian of the function $f(\cdot)$ with respect to the noise variables evaluated at the last estimation of the state mean $\boldsymbol{\mu}_{k-1}$, with the current input $\mathbf{u}_k$ and assuming no noises.

Similarly, the measurement is given by a function $h(\cdot)$ of the state, some noises, and, to be general, some deterministic input in a q elements vector $\mathbf{v}$:

$$\mathbf{z}_k = h(\mathbf{x}_{k-1}, \mathbf{v}_k, \boldsymbol{\delta}_k). \tag{4.84}$$

The update step can be performed as

$$\begin{aligned}
\mathbf{S} &\simeq \mathbf{H}_\mathbf{x}\hat{\boldsymbol{\Sigma}}_k\mathbf{H}_\mathbf{x}^T + \mathbf{H}_\delta\boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}\mathbf{H}_{\boldsymbol{\delta}}^T, && (4.85)\\
\mathbf{K} &= \hat{\boldsymbol{\Sigma}}_k\mathbf{H}_\mathbf{x}^T\mathbf{S}^{-1}, && (4.86)\\
\hat{\mathbf{h}} &= h(\hat{\boldsymbol{\mu}}_k, \mathbf{v}_k, \boldsymbol{\delta}_k = \mathbf{0}), && (4.87)\\
\boldsymbol{\mu}_k &= \hat{\boldsymbol{\mu}}_k + \mathbf{K}(\mathbf{z}_k - \hat{\mathbf{h}}), && (4.88)\\
\boldsymbol{\Sigma}_k &= \hat{\boldsymbol{\Sigma}}_k - \mathbf{K}\mathbf{H}_\mathbf{x}\hat{\boldsymbol{\Sigma}}_k, && (4.89)
\end{aligned}$$

where

$$\mathbf{H}_\mathbf{x} = \left.\frac{\partial h(\mathbf{x}, \mathbf{v}, \boldsymbol{\delta})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\boldsymbol{\mu}}_k, \mathbf{v}=\mathbf{v}_k, \boldsymbol{\delta}=\mathbf{0}}, \tag{4.90}$$

is the Jacobian of the function $h(\cdot)$ with respect to the state variables evaluated at the last estimation of the state mean $\hat{\boldsymbol{\mu}}_{k-1}$, with the current input $\mathbf{v}_k$, assuming no noises, and

$$\mathbf{H}_\delta = \left.\frac{\partial f(\mathbf{x}, \mathbf{v}, \boldsymbol{\delta})}{\partial \boldsymbol{\delta}}\right|_{\mathbf{x}=\hat{\boldsymbol{\mu}}_k, \mathbf{v}=\mathbf{v}_k, \boldsymbol{\delta}=\mathbf{0}}. \tag{4.91}$$

is the Jacobian of the function $h(\cdot)$ with respect to the noise variables evaluated at the last estimation of the state mean $\hat{\boldsymbol{\mu}}_{k-1}$, with the current input $\mathbf{v}_k$, assuming no noises.

The complete algorithm that performs an estimation step in the Extended Kalman Filter is sketched in Algorithm 3.

Algorithm 3 $[\boldsymbol{\mu}_k, \Sigma_k]$ = EKF($\boldsymbol{\mu}_{k-1}$, $\boldsymbol{\Sigma}_{k-1}$, $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{z}_k$, $f(\cdot)$, $h(\cdot)$, $\mathbf{F}_\mathbf{x}$, $\mathbf{F}_{\boldsymbol{\eta}}$, $\mathbf{H}_\mathbf{x}$, $\mathbf{F}_{\boldsymbol{\delta}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}$, $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}$)

```
1: $\hat{\boldsymbol{\mu}}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k = \mathbf{0})$
2: $\hat{\boldsymbol{\Sigma}}_k = \mathbf{F}_\mathbf{x}\boldsymbol{\Sigma}_{k-1}\mathbf{F}_\mathbf{x}^T + \mathbf{F}_{\boldsymbol{\eta}}\boldsymbol{\Sigma}_{\boldsymbol{\eta}_k}\mathbf{F}_{\boldsymbol{\eta}}^T$
3: $\mathbf{S} = \mathbf{H}_\mathbf{x}\hat{\boldsymbol{\Sigma}}_k\mathbf{H}_\mathbf{x}^T + \mathbf{H}_\delta\boldsymbol{\Sigma}_{\boldsymbol{\delta}_k}\mathbf{H}_{\boldsymbol{\delta}}^T$
4: $\mathbf{K} = \hat{\boldsymbol{\Sigma}}_k\mathbf{H}_\mathbf{x}^T\mathbf{S}^{-1}$
5: $\hat{\mathbf{h}} = h(\hat{\boldsymbol{\mu}}_k, \mathbf{v}_k, \boldsymbol{\delta}_k = \mathbf{0})$
6: $\boldsymbol{\mu}_k = \hat{\boldsymbol{\mu}}_k + \mathbf{K}(\mathbf{z}_k - \hat{\mathbf{h}})$
7: $\boldsymbol{\Sigma}_k = \hat{\boldsymbol{\Sigma}}_k - \mathbf{K}\mathbf{H}_\mathbf{x}\hat{\boldsymbol{\Sigma}}_k$
```

4.3 Non Linear Least Squares Optimization

The optimization is the choice of an input value from within an allowed set that maximize or minimize a real function. In particular, here two *Non Linear Least Squares*

(NLLS) techniques are presented: the *Gauss-Newton* method and a variation of it named *Levenberg-Marquardt* algorithm. A good introduction to non linear least squares methods can be found in [57], while a more detailed analysis can be found in [33] and [22]. For The method of least squares is a standard approach to the approximate solution of linear overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. The overall solution minimizes the sum of the squares of the errors made in the results of every single equation. Non-linear least squares is the form of least squares analysis which is used to fit a set of m observations with a model that is non-linear in n unknown parameters with $m > n$.

Let consider the problem of finding the minimum of a function with this form:

$$\mathbf{F}(\mathbf{x}) = \mathbf{e}(\mathbf{x})^T \mathbf{\Omega}\, \mathbf{e}(\mathbf{x}), \tag{4.92}$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ \mathbf{F}(\mathbf{x}), \tag{4.93}$$

where

- $\mathbf{x}$ is a n elements vector of parameters to estimate,
- $\mathbf{e}(\cdot)$ is function $\mathbb{R}^n \to \mathbb{R}^m$ with $m \geq n$,
- $\mathbf{\Omega}$ is a $m \times m$ information matrix (i.e., the inverse of a covariance matrix) that acts as weight on $\mathbf{e}(\cdot)$ elements.

Supposing that an initial estimation of $\mathbf{x}$ parameters sufficiently close to the optimum is known, the first order Taylor approximation of $\mathbf{F}(\mathbf{x} + \mathbf{\Delta x})$ can be considered in order to find the minimum point with an iterative procedure:

$$\begin{aligned}
\mathbf{F}(\mathbf{x}+\mathbf{\Delta x}) &= \mathbf{e}(\mathbf{x}+\mathbf{\Delta x})^T \mathbf{\Omega}\, \mathbf{e}(\mathbf{x}+\mathbf{\Delta x}) && (4.94)\\
&\simeq [\mathbf{e}(\mathbf{x}) + \mathbf{J\Delta x}]^T \mathbf{\Omega}\ [\mathbf{e}(\mathbf{x}) + \mathbf{J\Delta x}] && (4.95)\\
&= \mathbf{e}(\mathbf{x})^T \mathbf{\Omega}\, \mathbf{e}(\mathbf{x}) + \mathbf{e}(\mathbf{x})^T \mathbf{\Omega J \Delta x}\ + && (4.96)\\
&\quad \mathbf{\Delta x}^T \mathbf{J}^T \mathbf{\Omega e}(\mathbf{x}) + \mathbf{\Delta x}^T \mathbf{J}^T \mathbf{\Omega J \Delta x} && (4.97)\\
&= \underbrace{\mathbf{e}(\mathbf{x})^T \mathbf{\Omega}\, \mathbf{e}(\mathbf{x})}_{c} + 2\underbrace{\mathbf{e}(\mathbf{x})^T \mathbf{\Omega J}}_{\mathbf{b}^T} \mathbf{\Delta x}\ + \mathbf{\Delta x}^T \underbrace{\mathbf{J}^T \mathbf{\Omega J}}_{\mathbf{H}} \mathbf{\Delta x} && (4.98)\\
&= c + 2\mathbf{b}^T \mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H \Delta x}, && (4.99)
\end{aligned}$$

where

$$\mathbf{J} = \left.\frac{\partial \mathbf{e}(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right|_{\tilde{\mathbf{x}}=\mathbf{x}}, \tag{4.100}$$

is the $m \times n$ Jacobian matrix of the error function evaluated at the current value of the $\mathbf{x}$ parameters and $\mathbf{H}$ is the information matrix of the system. The formula in Equation 4.99 is a quadratic form and can be minimized by solving the linear system

$$\mathbf{H \Delta x}^* = -\mathbf{b}, \tag{4.101}$$

through Cholesky decomposition, QR decomposition or with iterative methods. A new point for the linearization is obtained by adding the solution $\mathbf{\Delta x}^*$ to the current parameters vector $\mathbf{x}$

$$\mathbf{x} = \mathbf{x} + \mathbf{\Delta x}^*. \tag{4.102}$$

The popular Gauss-Newton minimization algorithm iterates, starting with a initial guess $\mathbf{x}_0$, through the following steps

1. the linearization shown in Equation 4.99,
2. the computation of the solution of Equation 4.101,
3. the update of the estimation of Equation 4.102,

until a given termination criterion is reached. Some common stop criterion are

- Maximum number of iteration reached
- Stability in the update, i.e., $\|\mathbf{x}\| < \epsilon_{\mathbf{x}}$
- Stability in the function value, i.e., $|\mathbf{F}(\mathbf{x} + \mathbf{\Delta x}) - \mathbf{F}(\mathbf{x})| < \epsilon_{\mathbf{F}}$

Algorithm 4 $\mathbf{x}^* =$ Gauss-Newton($\mathbf{x}_0$, $\mathbf{e}(\tilde{\mathbf{x}})$, $\frac{\partial \mathbf{e}(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}$, $\mathbf{\Omega}$)

```
1: $\mathbf{x} = \mathbf{x}_0$
2: repeat
3:     $\mathbf{J} = \left.\frac{\partial \mathbf{e}(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right|_{\tilde{\mathbf{x}}=\mathbf{x}}$
4:     $\mathbf{b} = \mathbf{e}(\mathbf{x})^T \mathbf{\Omega} \mathbf{J}$
5:     $\mathbf{H} = \mathbf{J}^T \mathbf{\Omega} \mathbf{J}$
6:     solve $\mathbf{H}\mathbf{\Delta x}^* = -\mathbf{b}$ for $\mathbf{\Delta x}^*$
7:     $\mathbf{x} = \mathbf{x} + \mathbf{\Delta x}^*$
8: until termination criterion
9: $\mathbf{x}^* = \mathbf{x}$
```

The complete Gauss-Newton minimization algorithm is sketched in Algorithm 4. It can be shown (see [22]) that the increment $\mathbf{\Delta x}$ is a descent direction and, if the algorithm converges, then the limit is a stationary point for $\mathbf{F}$. Convergence is guaranteed if the region in which $\mathbf{F}(\mathbf{x}) < \mathbf{F}(\mathbf{x}_0)$ is bounded and $\mathbf{J}$ has full rank in all steps. Notice that if there is a linear dependency between columns of $\mathbf{J}$, the $\mathbf{H}$ matrix becomes singular. Convergence rate can approach quadratic, but in general we must expect linear convergence.

4.3.1 Non-Euclidean Spaces

The considered Taylor approximation of Equation 4.99 and the update step of Equation 4.102 assume that the space of parameters vector $\mathbf{x}$ is Euclidean. In general this is not true, thus a common approach is to express the increments $\mathbf{\Delta x}$ in a space different from $\mathbf{x}$. Moreover, there are cases in which it is necessary to use an over-parametrized representation of space to avoid singularities (e.g., the case of quaternions, which use 4 elements to represent rotations in 3D) and this causes close to singular or singular $\mathbf{H}$ matrix, thus preventing from convergence of the minimization.

A solution that copes with non-Euclidean spaces is to consider $\mathbf{\Delta x}$ as a perturbation vector around $\mathbf{x}$ such that the composition of the displacement is not simply the summation of the vectors, but a generic function expressed by the operator $\boxplus$. Thus, Equation 4.102 will be changed in

$$\mathbf{x} \;=\; \mathbf{x} \boxplus \mathbf{\Delta x}, \tag{4.103}$$

where $\mathbf{\Delta x}$ is a vector of r elements, with r potentially different from n. Thus, $\boxplus$ is a functional operator $\mathbb{R}^n \times \mathbb{R}^r \rightarrow \mathbb{R}^n$. This implies that the Equation 4.99 becomes

$$\begin{aligned} \mathbf{F}(\mathbf{x} \boxplus \mathbf{\Delta x}) &= \mathbf{e}\left(\mathbf{x} \boxplus \mathbf{\Delta x}\right)^T \mathbf{\Omega}\, \mathbf{e}\left(\mathbf{x} \boxplus \mathbf{\Delta x}\right) && (4.104) \\ &\simeq \left[\mathbf{e}\left(\mathbf{x}\right) + \mathbf{J}\mathbf{\Delta x}\right]^T \mathbf{\Omega}\, \left[\mathbf{e}\left(\mathbf{x}\right) + \mathbf{J}\mathbf{\Delta x}\right] && (4.105) \\ & && (4.106) \end{aligned}$$

where

$$\mathbf{J} \;=\; \left.\frac{\partial \mathbf{e}\left(\tilde{\mathbf{x}} \boxplus \mathbf{\Delta x}\right)}{\partial \mathbf{\Delta x}}\right|_{\mathbf{\Delta x}=\mathbf{0},\, \tilde{\mathbf{x}}=\mathbf{x}} \tag{4.107}$$

is the $m \times r$ Jacobian matrix evaluated at the current estimation $\mathbf{x}$ considering a null increment $\mathbf{\Delta}x$.

4.3.2 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm introduces a scalar constant λ on the main diagonal of the $\mathbf{H}$ to control algorithm convergence, solving

$$\left(\mathbf{H} + \lambda \mathbf{I}\right) \mathbf{\Delta x}^* \;=\; -\mathbf{b} \tag{4.108}$$

instead of Equation 4.101. The higher is λ, the smallest is the step Δx. This is useful to control the step size on nonlinear surfaces by monitoring the error trend: if the error decreases with respect to the previous iteration, λ is decreased and a new iteration is performed, otherwise the previous solution is reverted and λ is increased. Moreover, the Levenberg-Marquardt is able to treat natively problems where the Hessian matrix $\mathbf{H}$ results to be singular, although this condition has to be avoided, since it is symptom of an improperly defined problem.

CHAPTER 5

Simultaneous Localization And Mapping

One of the most fundamental problems in mobile robotics, but not solely robotics, is the so called *Simultaneous Localization And Mapping (SLAM)* problem. It asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and to incrementally build a consistent map of this environment while simultaneously determining its location within this map. A solution to the SLAM problem has been seen as a "holy grail" for the mobile robotics community as it would make a robot truly autonomous, as stated in [26].

SLAM has been formulated and solved as a theoretical problem in a number of different forms and it has also been implemented in a number of different domains from indoor robots to outdoor, underwater, and airborne systems. At a theoretical and conceptual level, SLAM can now be considered a solved problem. However, there are still a large number of issues that need to be addressed in order to develop and deploy complete SLAM applications.

In this chapter we introduce the SLAM problem statement, comparing it with the "simpler" problems of localization and mapping. Then we give a short review of the SLAM history, citing some applications presented in the literature. Starting from the probabilistic formulation of the SLAM problem we describe one of the most studied approach to the SLAM: the on line EKF-SLAM algorithm. An alternative formulation is the Graph Formulation, that describes the problem as a optimization problem on a graph. The last section of this chapter describes a recent framework, i.e., the Conditional Independence Sub-Mapping SLAM, that can be applied to the EKF-SLAM algorithm to address large scale problems, i.e., problems with large maps and long explored paths.

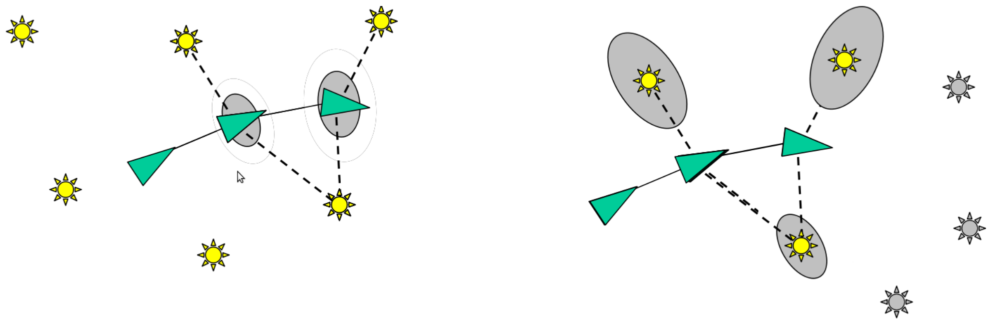


Figure 5.1: *An intuitive sketch of (a) localization problem, (b) mapping problem. Map elements (landmarks) are the stars, the triangle is the the robot; ellipses represent elements with uncertainties; gray landmarks are not mapped yet.*

5.1 SLAM Problem Statement

To introduce SLAM problem, it is useful to step back to the simpler problems of *localization* and *mapping*. The *localization* problem can be seen as the ability for a robot (or a generic mobile device) to answer to the question "where am I?", i.e., to localize itself with respect to a known map using proprioceptive information provided by one or more sensors. Let suppose that the robot cannot sense its pose directly (i.e., it does not have a sort of GPS system), but it can sense where some salient elements of the map (or *landmarks*) are. The localization problem can be seen as a problem of coordinate transformation: the map is known with respect to a global reference system, the observations are referenced to the actual robot position and the unknown is the robot position with respect to the global reference system. It should be noticed that localization problem solutions have to take in account noise in sensor measurements.

On the other hand, the *mapping* problem can be seen as the ability for a robot to answer to the question "how is the environment in which I operate?". This can be achieved by merging time by time the sensors observations to provide a complete map of the environment, assuming that the position of the robot is precisely known at each time step. The map building process has to take into account noise in sensor measurements in order to build a consistent and precise map with information integrated over time. Figure 5.1(a) and Figure 5.1(b) give respectively a quick view of the localization and mapping problems, the stars representing the landmarks in the map and the triangles representing the robot poses in time. Shaded ellipses around objects represent uncertainties: in localization uncertainties are in the robot pose, while in mapping they are on the landmark position.

The SLAM problem is the combination of the localization and mapping problem. It asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and to incrementally build a consistent map of this environment while simultaneously determining its location within this map. The SLAM problem can be formulated as the question *"How can a mobile robot operate in an a priori unknown environment and use only onboard sensors to simultaneously build a map of its workspace and use it to*

]

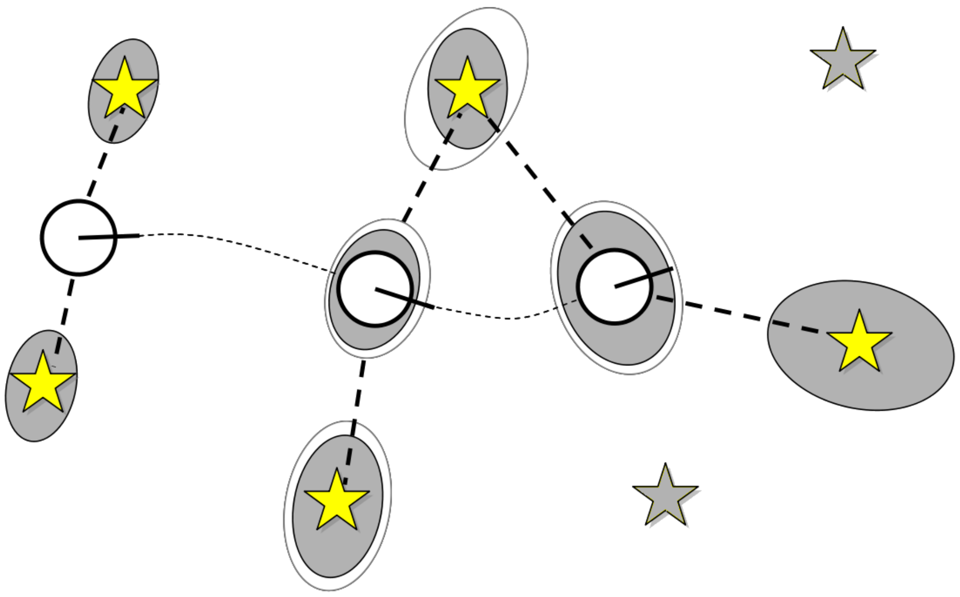


Figure 5.2: *An intuitive sketch of the SLAM problem. Map elements (landmarks) are the stars, the triangle is the the robot; ellipses represent elements with uncertainties; gray elements are unknown; gray landmarks are not mapped yet.*

navigate?"[1]. In SLAM the robot incrementally builds the map of the environment while simultaneously localizing itself in the map. The problem is sketched in Figure 5.2, where it is clearly shown that uncertainties are both on landmarks positions and robot poses. Moreover, robot pose uncertainty and map uncertainties are correlated, since the localization is performed on an uncertain map and the map in turn is build using an uncertain robot pose. Intuitively localization problem and mapping problem are "simpler" than the SLAM problem.

An underneath issue for all the three presented problems is the so called *data association* problem [1]. Considering for simplicity the localization problem, we state that sensors are able to produce some measurements of the surrounding landmarks. These measurements are useless for localization if they are not associated with specific landmarks in the map, e.g., it is useless to know that there is a wall at three meters in front of the robot without knowing if the wall is the northern or the southern of the room. The difficulty of the data association problem varies depending on the specific application. Let consider an example: if our robot is moving in a very simple environment with few landmarks distinguishable by color, the data association can be easily solved by the color association; more complex is the case in which all landmarks are not distinguishable, for instance when they are blank circles in the environment. The data association problem covers an important role in the definition of the algorithms for the solution of localization,

[1]As stated by Prof. Paul Newman, Principal Investigator of Mobile Robotics Group, Oxford, in an article appeared on the Issue 4 of *SOUE news*, the magazine of the Society of Oxford University Engineers (`http://www.soue.org.uk/souenews/issue4/mobilerobots.html`)

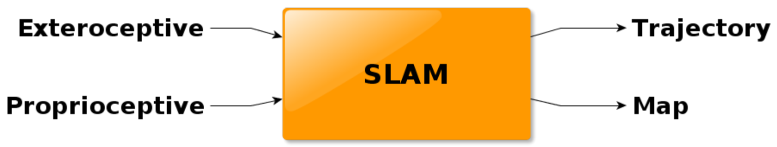


Figure 5.3: *The SLAM functional block with its inputs (both proprioceptive and exteroceptive) and outputs (the estimated trajectory and the estimated map).*

mapping and SLAM problems. Poor performance in data association prevents algorithms from obtaining a valid solution, and perfect data association is very uncommon in real applications. For the purpose of this work we consider that the data association is given but with the presence of some outliers. In particular we consider that the data association can be performed by some distinguishable characteristics of the landmark perceived by the measurement apparatuses but this can introduces spurious results that have to be detected by an outliers rejection system.

In summary, at this stage we consider the SLAM as a functional blocks with inputs and outputs, reported in Figure 5.3. It receives as input the *exteroceptive* measurements, i.e., the environmental measurements acquired by sensors, and the *proprioceptive* measurements, i.e., the informations that the robot, or the moving apparatus, knows about its own movements, e.g., odometric informations, inertial measurements or control inputs. The proprioceptive measurements can be absent, resulting in a SLAM approach that is based only on environmental observations. The output of the SLAM system is the estimation robot trajectory at each time step and the map estimation. The map has to be represented for the purpose of this work with geometrical entities, e.g., points, lines or planes.

5.2 SLAM History and Applications

The genesis of the probabilistic SLAM problem is set, according to [26], in the 1986 IEEE Robotics and Automation Conference held in San Francisco, California, where a group of researchers had been looking at applying probabilistic and estimation theoretic methods to localization and mapping problems. In few years, a number of key papers were produced and they laid the foundations for successive investigations. Among them we can cite [87], [25], [88] and [51]. These works put in evidence that the map of the environment, incrementally build during the exploration as a set of landmark positions, is fully correlated with the robot position. Consequently, the simultaneous localization and mapping problem, also know at early stages as *concurrent localization and mapping (CML)*, has to be addressed by the continuous estimation of the joint state composed by the vehicle pose and the landmarks, resulting in a huge state vector that grows with the number of observed landmarks. The computational complexity of the problem was intractable, thus the most of the following works concentrated in approximating the problem, assuming or forcing the correlations to be deleted. The most important works in this period were based on sonar sensors, as in [52] and [80].

A fundamental step was performed when it was demonstrated that approximations in the estimation process prevent the solution from convergence. This was definitely clear in [24], where the acronym SLAM was coined. After this, several research groups all around the world have deeply investigated the SLAM problem, proposing applications in indoor, outdoor and underwater environments. Consequently, numerous dedicated session in conferences and symposium have been attracting an exponential number of researchers and student. Similarly, summer schools and courses on SLAM techniques have been on the crest of the wave up to now.

SLAM applications now exists in a wide variety of domains, including indoor environment [12] [19], outdoor [28] [36], aerial [45] and underwater [27] [71]. Various exteroceptive sensors types have been used, such as sonars [12] [71], laser scanners [18] and, more recently, vision [19]. In particular, the latter is the context in which this work takes place, thus more details and references related to Visual SLAM will be given in this and in the next chapters.

5.3 Probabilistic Formulation

The desired output of a SLAM system, i.e., the solution of the SLAM problem, is the estimation of the robot trajectory and the map of the explored environment. Estimation has to take in account noise in sensor measurements, thus a SLAM problem can be described in terms of probability distributions. Naming $\mathbf{\Gamma}$ the robot pose (i.e., the position and orientation of the robot) expressed with respect to a "world" reference frame, the complete trajectory of the robot can be described by the sequence of random variables $\mathbf{\Gamma}_{1:T} = \{\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \dots, \mathbf{\Gamma}_{T-1}, \mathbf{\Gamma}_T\}$. To be general, we assume that the robot knows its movement at each time step, i.e., it has a set of inputs represented by the random variables sequence $\mathbf{u}_{1:T} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{T-1}, \mathbf{u}_T\}$. These input are usually represented by readings of odometric sensors, by measurements of an Inertial Measurement Unit (IMU) or by controls given to the motors, but in some cases they can be absent at all. The environment is represented by a random variable $\mathbf{m}$ representing the map. The map $\mathbf{m}$, in turn, is represented by a set of random variables $\mathbf{y}_i$, each of them representing a single landmark of the environment $\mathbf{m} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$. It can be noticed that the map is independent from time, i.e., we are assuming that the map, namely each landmark, is static, thus it is not necessary to estimate $\mathbf{m}_t$ for each time step t. This is not the unique possible formulation for the SLAM problem. For instance, it is possible to formulate a more complex problem discarding the assumption on static landmarks (e.g., in [89]).

Solving the so called *full SLAM problem* consist in the estimation of the posterior probability

$$\boldsymbol{p}(\mathbf{\Gamma}_{1:T}, \mathbf{m} | \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{\Gamma}_0), \tag{5.1}$$

i.e., the joint probability of the trajectory and the map given all environmental measurements, input and the initial pose $\mathbf{\Gamma}_0$. The initial pose is arbitrary and can be chosen freely; changes in the initial pose result in a roto-traslated solution, therefore it is common to assume the start pose in the origin and to omit the term $\mathbf{\Gamma}_0$ from the probabilistic formulation to ease notation.

The so called *on-line SLAM problem* marginalize out all past poses of the trajectory, focusing on the estimation of the last (or current) pose and the map:

$$\boldsymbol{p}(\mathbf{\Gamma}_T, \mathbf{m} | \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{\Gamma}_0), \tag{5.2}$$

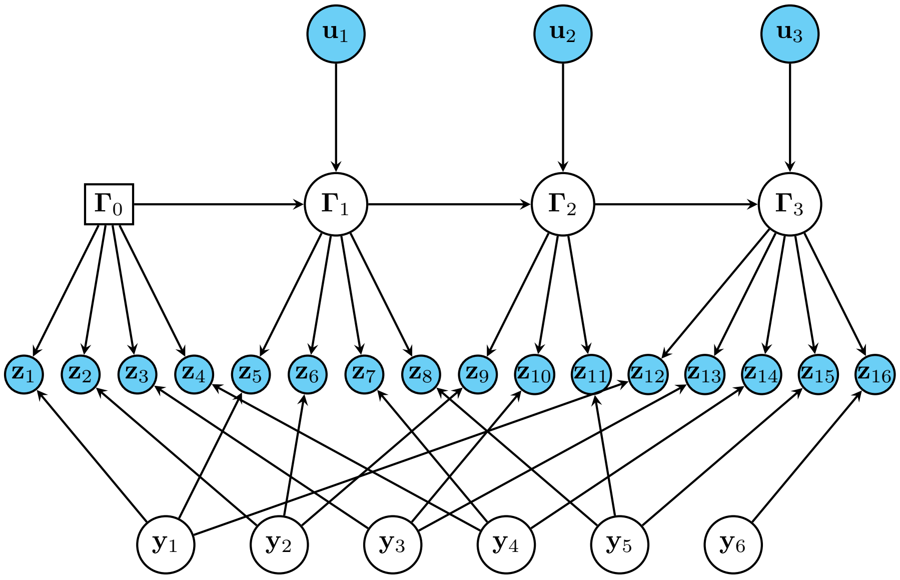

$\mathbf{u}_1$
$\mathbf{u}_2$
$\mathbf{u}_3$
$\mathbf{\Gamma}_0$
$\mathbf{\Gamma}_1$
$\mathbf{\Gamma}_2$
$\mathbf{\Gamma}_3$
$\mathbf{z}_1$ $\mathbf{z}_2$ $\mathbf{z}_3$ $\mathbf{z}_4$ $\mathbf{z}_5$ $\mathbf{z}_6$ $\mathbf{z}_7$ $\mathbf{z}_8$ $\mathbf{z}_9$ $\mathbf{z}_{10}$ $\mathbf{z}_{11}$ $\mathbf{z}_{12}$ $\mathbf{z}_{13}$ $\mathbf{z}_{14}$ $\mathbf{z}_{15}$ $\mathbf{z}_{16}$
$\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$ $\mathbf{y}_4$ $\mathbf{y}_5$ $\mathbf{y}_6$

Figure 5.4: *Dynamic Bayesian Network (DBN) of the full SLAM problem.*

and, considering the Markov assumption on the state completeness we can reformulate Equation 5.2 in a recursive form:

$$p(\mathbf{\Gamma}_T, \mathbf{m} | \mathbf{z}_T, \mathbf{u}_T, \mathbf{\Gamma}_{T-1}). \tag{5.3}$$

The SLAM problem can be easily and conveniently represented by a Dynamic Bayesian Network (DBN) too. The DBN for the full SLAM problem, depicted in Figure 5.4, presents one node for each random variable involved in the estimation process and the directed edges between nodes represent the conditional dependences. Cyan nodes represents the observed variables (sensor readings and landmarks measurements), white nodes are the hidden variables that are the subject of the estimation. The node $\mathbf{\Gamma}_0$ is square to enhance its particular role, since it is not involved in the estimation.

The connectivity of the DBN follows the Recursive Bayesian Filtering pattern introduced in Section 4.2.1. The *state transition probability* is expressed by

$$p(\mathbf{\Gamma}_k | \mathbf{\Gamma}_{k-1}, \mathbf{u}_t), \tag{5.4}$$

and it is represented by the two edges entering in a $\mathbf{\Gamma}$ node. The *measurement probability*

$$p(\mathbf{z}_i | \mathbf{\Gamma}_k, \mathbf{y}_j) \tag{5.5}$$

expresses the probability of the i-th measure $\mathbf{z}_i$ obtained when the robot is located at $\mathbf{\Gamma}_k$ and sensors are measuring the j-th landmark. In the DBN formulated above, measurements are represented by the arrows entering in the $\mathbf{z}$ nodes.

5.4 EKF Based On-Line SLAM

In the previous section we introduced the SLAM problem in the realm of recursive Bayesian filtering. Here, we focus on the use of the Extended Kalman Filter (EKF) presented in Section 4.2.3 to solve the on-line SLAM problem, called EKF-SLAM [102]. This is a specific application of the EKF to recursively estimate, from environmental measurements, the joint distribution of the actual robot pose and landmarks in the environment as a single multidimensional Gaussian distribution. In the reminder of this section, the basic techniques that allow to treat the EKF-SLAM problem in real time, taking advantage from the particular structure of the problem are presented together with the definition of the state vector, the prediction step equations and the measurement equations. All the involved equations are introduced in a general form, without any reference to the specific SLAM implementations based on specific sensorial apparatuses.

5.4.1 State Vector

In EKF-SLAM, the state of the filter $\mathbf{x}_k$, at step k, is composed by the current robot pose $\mathbf{\Gamma}_k$, including both translation and orientation, in a world reference frame and the map $\mathbf{m}_k$ represented by a set of landmarks $\mathbf{m}_k = \{\mathbf{y}_{ik}\}$. Each landmark encodes in some way the position and eventually other characteristics of a single landmark of the environment with respect to the world reference frame. Some other parameters $\mathbf{\Lambda}_t$, describing robot motion (e.g., tangential and rotational speed or accelerations), could be part of the EKF state as well and they could be represented with respect to the robot reference frame or the world reference frame, in accordance with the specific *motion model* of the system. This formulation of the EKF SLAM problem is the most intuitive, it is called the *world centric* SLAM; it has some nice properties that will be introduced in the next paragraphs. An alternative formulation is the so called *robocentric SLAM*, that refers landmarks to the last robot position. We do not consider this formulation hereafter, although it has good properties clearly showed in [16] [61] at the cost of an increased computational complexity.

The complete state of the filter is thus represented as:

$$\mathbf{x}_t = \left[\begin{array}{cccccc} \mathbf{\Gamma}_k^T & \mathbf{\Lambda}_k^T & \mathbf{y}_{1k}^T & \mathbf{y}_{2k}^T & \dots & \mathbf{y}_{nk}^T \end{array} \right]^T, \tag{5.6}$$

and the EKF-SLAM algorithm follows the steps described in Algorithm 5 to update recursively the state mean estimation $\boldsymbol{\mu}_k$, together with its covariance matrix $\mathbf{\Sigma}_k$. In the following, this algorithm will be explained in details.

5.4.2 Initialization

Unless we have some prior information, at $t = 0$ we start with an empty map and the initial robot pose is usually considered as the origin of the world (let name it $\bar{\mathbf{\Gamma}}_0^T$) with no uncertainty. Furthermore, if we are considering some parameters to describe the robot motion, we can initialize them as zero or with a specific mean value (name it $\bar{\mathbf{\Lambda}}_0$), with a non zero initial covariance ($\mathbf{\Sigma}_{\mathbf{\Lambda}_0}$):

$$\boldsymbol{\mu}_0 = \left[\bar{\mathbf{\Gamma}}_0^T,\ \bar{\mathbf{\Lambda}}_0^T\right], \quad \mathbf{\Sigma}_0 = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{\Sigma}_{\mathbf{\Lambda}_0} \end{bmatrix}. \tag{5.7}$$

Algorithm 5 EKF-SLAM

```
 1: # *Initialization*
 2: $k \leftarrow 0$, $\boldsymbol{\mu}_k \leftarrow \begin{bmatrix} \bar{\boldsymbol{\Gamma}}_0 \\ \bar{\boldsymbol{\Lambda}}_0 \end{bmatrix}$, $\boldsymbol{\Sigma}_k \leftarrow \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_0} \end{bmatrix}$.
 3: loop
 4:    # *Prediction step*
 5:    $k \leftarrow k+1$
 6:    $\hat{\boldsymbol{\mu}}_k = \begin{bmatrix} f_{\mathbf{D}}(\bar{\boldsymbol{\Gamma}}_{k-1}, \bar{\boldsymbol{\Lambda}}_{k-1}, \bar{\mathbf{u}}_k, \boldsymbol{\eta}_k = \mathbf{0}) \\ \bar{\mathbf{m}}_{k-1} \end{bmatrix}$
 7:    $\hat{\boldsymbol{\Sigma}}_k = \begin{bmatrix} \mathbf{F}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{DD}k-1}\mathbf{F}_{\mathbf{D}}^T + \mathbf{F}_{\mathbf{D}\eta}\boldsymbol{\Sigma}_{\boldsymbol{\eta}}\mathbf{F}_{\mathbf{D}\eta}^T & \mathbf{F}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{Dm}k-1} \\ \boldsymbol{\Sigma}_{\mathbf{Dm}k-1}^T\mathbf{F}_{\mathbf{D}}^T & \boldsymbol{\Sigma}_{\mathbf{mm}k-1} \end{bmatrix}$
 8:    # *Measurement step*
 9:    $\mathbf{h} \leftarrow \varnothing$, $\mathbf{z} \leftarrow \varnothing$, $\mathbf{H} \leftarrow \varnothing$, $\mathbf{R} \leftarrow \varnothing$
10:    for all $\mathbf{y}_{i_k}$ do
11:       $\mathbf{h}_i = h_i(\boldsymbol{\mu}_k, \mathbf{v}_i, \boldsymbol{\delta}_i = 0)$
12:       if $\mathbf{z}_i$ is compatible with $\mathbf{h}_i$ then
13:          $\mathbf{h} \leftarrow \begin{bmatrix} \mathbf{h} \\ \mathbf{h}_i \end{bmatrix}$, $\mathbf{z} \leftarrow \begin{bmatrix} \mathbf{z} \\ \mathbf{z}_i \end{bmatrix}$, $\mathbf{H} \leftarrow \begin{bmatrix} \mathbf{H} \\ \mathbf{H}_{i\mathbf{x}} \end{bmatrix}$, $\mathbf{R} \leftarrow \mathrm{diag}(\mathbf{R},\ \mathbf{H}_{i\boldsymbol{\delta}}\boldsymbol{\Sigma}_{\boldsymbol{\delta}}\mathbf{H}_{i\boldsymbol{\delta}})$
14:       end if
15:    end for
16:    # *Update step*
17:    if h $\neq \varnothing$ then
18:       $\mathbf{S} \leftarrow \mathbf{H}\hat{\boldsymbol{\Sigma}}_k\mathbf{H}^T + \mathbf{R}$
19:       $\mathbf{K} \leftarrow \hat{\boldsymbol{\Sigma}}_k\mathbf{H}^T\mathbf{S}^{-1}$
20:       $\boldsymbol{\mu}_k \leftarrow \hat{\boldsymbol{\mu}}_k + \mathbf{K}(\mathbf{z} - \mathbf{h})$
21:       $\boldsymbol{\Sigma}_k \leftarrow \hat{\boldsymbol{\Sigma}}_k - \mathbf{KSK}^T$
22:    else
23:       $\boldsymbol{\mu}_k \leftarrow \hat{\boldsymbol{\mu}}_k$, $\boldsymbol{\Sigma}_k \leftarrow \hat{\boldsymbol{\Sigma}}_k$
24:    end if
25:    # *New landmark addition*
26:    for all $new\_measurement$ s do
27:       $\bar{\mathbf{y}}_{new} = g(\boldsymbol{\mu}_k, \bar{\mathrm{s}}, \boldsymbol{\xi} = \mathbf{0})$
28:       $\boldsymbol{\mu}_k \leftarrow \begin{bmatrix} \boldsymbol{\mu}_k \\ \bar{\mathbf{y}}_{new} \end{bmatrix}$, $\boldsymbol{\Sigma}_k \leftarrow \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{DD}k} & \boldsymbol{\Sigma}_{\mathbf{Dm}k} & \boldsymbol{\Sigma}_{\mathbf{DD}k}\mathbf{G}_{\mathbf{D}}^T \\ \boldsymbol{\Sigma}_{\mathbf{Dm}k}^T & \boldsymbol{\Sigma}_{\mathbf{mm}k} & \boldsymbol{\Sigma}_{\mathbf{Dm}k}^T\mathbf{G}_{\mathbf{D}}^T \\ \mathbf{G}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{DD}k} & \mathbf{G}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{Dm}k} & \mathbf{G}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{DD}k}\mathbf{G}_{\mathbf{D}}^T + \mathbf{G}_{\xi}\boldsymbol{\Sigma}_{\xi}\mathbf{G}_{\xi}^T \end{bmatrix}$
29:    end for
30: end loop
```

5.4.3 Prediction Step

At each iteration, we perform the prediction step of the Extended Kalman filter; this step aims at the prediction of the next system state, using prior state values and an optional control input $\mathbf{u}_k$ while assuming a non additive Gaussian noise $\boldsymbol{\eta}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\eta)$ perturbates the motion. With reference to Equation 4.78, the state transition has to be specified in terms of function $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k)$. Since the state vector is partitioned in robot pose and landmarks, this state transition function can be decomposed as

$$\begin{bmatrix} \boldsymbol{\Gamma}_k \\ \boldsymbol{\Lambda}_k \end{bmatrix} = f_{\mathbf{D}}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{5.8}$$

$$\begin{bmatrix} \mathbf{y}_{1k} \\ \mathbf{y}_{2k} \\ \vdots \\ \mathbf{y}_{nk} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{1k-1} \\ \mathbf{y}_{2k-1} \\ \vdots \\ \mathbf{y}_{nk-1} \end{bmatrix}; \tag{5.9}$$

this comes from the fact that landmarks are assumed to be static, i.e., they do not change location in time, while robot is moving according to a specific motion function.

To update the state covariance matrix we use the Jacobian matrices $\mathbf{F}_\mathbf{x} = \partial f(\mathbf{x}, \mathbf{u}, \boldsymbol{\eta})/\partial \mathbf{x}$ and $\mathbf{F}_{\boldsymbol{\eta}} = \partial f(\mathbf{x}, \mathbf{u}, \boldsymbol{\eta})/\partial \boldsymbol{\eta}$ evaluated at the current estimation and input and with zero noise (see Equation 4.82 and Equation 4.83). Thanks to the form of the $f(\cdot)$ function, these matrices result

$$\mathbf{F}_\mathbf{x} = \begin{bmatrix} \frac{\partial f_{\mathbf{D}}(\cdots)}{\partial(\boldsymbol{\Gamma}, \boldsymbol{\Lambda})} & 0 \\ 0 & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathbf{D}} & 0 \\ 0 & \mathbf{I} \end{bmatrix}, \qquad \mathbf{F}_{\boldsymbol{\eta}} = \begin{bmatrix} \frac{\partial f_{\mathbf{D}}(\cdots)}{\partial \boldsymbol{\eta}} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathbf{D}\eta} \\ 0 \end{bmatrix}. \tag{5.10}$$

Exploitation of the sparsity of the Jacobian matrix allows to simplify the prediction step complexity of the EKF algorithm. In particular, let consider the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ split in two part , where the first part contains the dynamic part of the system $\mathbf{D} = [\boldsymbol{\Gamma}^T, \boldsymbol{\Lambda}^T]^T$ and the second part contains the map $\mathbf{m}$ (i.e., all the landmarks). The mean vector and the covariance matrix results

$$\boldsymbol{\mu} = \begin{bmatrix} \bar{\mathbf{D}} \\ \bar{\mathbf{m}} \end{bmatrix}, \tag{5.11}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{DD}} & \boldsymbol{\Sigma}_{\mathbf{Dm}} \\ \boldsymbol{\Sigma}_{\mathbf{Dm}}^T & \boldsymbol{\Sigma}_{\mathbf{mm}} \end{bmatrix}. \tag{5.12}$$

where the over bar indicates that we are referring to the mean values of a variable. The prediction step performed by Equation 4.80 and Equation 4.81 results

$$\hat{\boldsymbol{\mu}}_k = \begin{bmatrix} \hat{\bar{\mathbf{D}}}_k \\ \hat{\bar{\mathbf{m}}}_k \end{bmatrix} = \begin{bmatrix} f_{\mathbf{D}}(\bar{\boldsymbol{\Gamma}}_{k-1}, \bar{\boldsymbol{\Lambda}}_{k-1}, \bar{\mathbf{u}}_k, \boldsymbol{\eta}_k = \mathbf{0}) \\ \bar{\mathbf{m}}_{k-1} \end{bmatrix}, \tag{5.13}$$

$$\hat{\boldsymbol{\Sigma}}_k = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_{\mathbf{DD}k} & \hat{\boldsymbol{\Sigma}}_{\mathbf{Dm}k} \\ \hat{\boldsymbol{\Sigma}}_{\mathbf{Dm}k}^T & \hat{\boldsymbol{\Sigma}}_{\mathbf{mm}k} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathbf{D}} \boldsymbol{\Sigma}_{\mathbf{DD}k-1} \mathbf{F}_{\mathbf{D}}^T + \mathbf{F}_{\mathbf{D}\eta} \boldsymbol{\Sigma}_\eta \mathbf{F}_{\mathbf{D}\eta}^T & \mathbf{F}_{\mathbf{D}} \boldsymbol{\Sigma}_{\mathbf{Dm}k-1} \\ \boldsymbol{\Sigma}_{\mathbf{Dm}k-1}^T \mathbf{F}_{\mathbf{D}}^T & \boldsymbol{\Sigma}_{\mathbf{mm}k-1} \end{bmatrix} \tag{5.14}$$

It can be noticed that a the second part of the mean vector and the bottom right block of the covariance matrix remains unchanged by the prediction step, thanks to the special form of the state transition function. Taking in account this special structure it allows to reduce the computational complexity of the prediction step from $O(n^3)$ to $O(n)$, being n the number of landmarks, that are usually the biggest part of the state vector.

5.4.4 Landmark Addition

In SLAM we aim at the online construction of the environment map, thus we need to enlarge the filter state each time a new landmark is perceived (see steps 27-28 of Algorithm 5). When a new landmark $\mathbf{y}_{new}$ is created, its initialization is computed as a function of the actual state vector and some information $\mathbf{s}$ that represents the first measure of the new landmark perturbed by Gaussian noise $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\xi}})$, which is, in general non additive:

$$\mathbf{y}_{new} = g(\mathbf{x}, \mathbf{s}, \xi). \tag{5.15}$$

This new landmark is added to the filter by enlarging the state vector

$$\mathbf{x}_k^* = [\mathbf{x}_k^T \quad \mathbf{y}_{new}^T]^T, \tag{5.16}$$

so, the mean vector results

$$\boldsymbol{\mu}_k^* = [\boldsymbol{\mu}_k^T \quad \bar{\mathbf{y}}_{new}^T]^T, \tag{5.17}$$

with

$$\bar{\mathbf{y}}_{new} = g(\boldsymbol{\mu}_k, \bar{\mathbf{s}}, \boldsymbol{\xi} = \mathbf{0}). \tag{5.18}$$

The Jacobians of the $g(\cdot)$ function, evaluated at the current mean values of the state vector and input, and not considering the noise, are

$$\mathbf{G}_{\mathbf{x}} = \left.\frac{\partial g(\cdots)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}_k, \mathbf{s}=\bar{\mathbf{s}}, \boldsymbol{\xi}=\mathbf{0}} \qquad \mathbf{G}_{\boldsymbol{\xi}} = \left.\frac{\partial g(\cdots)}{\partial \boldsymbol{\xi}}\right|_{\mathbf{x}=\boldsymbol{\mu}_k, \mathbf{s}=\bar{\mathbf{s}}, \boldsymbol{\xi}=\mathbf{0}}. \tag{5.19}$$

To update the covariance matrix it is possible to consider the addition step as a sort of prediction step that modifies the state vector by adding the new landmark, while it leaves unchanged all the dynamic part and the existing map of the state vector. Following a procedure similar to the one sketched in Section 5.4.3, the new covariance matrix results

$$\begin{aligned} \boldsymbol{\Sigma}_k^* &= \begin{bmatrix} \mathbf{I} \\ \mathbf{G}_{\mathbf{x}} \end{bmatrix} \boldsymbol{\Sigma}_k \begin{bmatrix} \mathbf{I} & \mathbf{G}_{\mathbf{x}}^T \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{G}_\xi \boldsymbol{\Sigma}_\xi \mathbf{G}_\xi^T \end{bmatrix}^T = & (5.20) \\ &= \begin{bmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{\Sigma}_k \mathbf{G}_{\mathbf{x}}^T \\ \mathbf{G}_{\mathbf{x}} \boldsymbol{\Sigma}_k & \mathbf{G}_{\mathbf{x}} \boldsymbol{\Sigma}_k \mathbf{G}_{\mathbf{x}}^T + \mathbf{G}_\xi \boldsymbol{\Sigma}_\xi \mathbf{G}_\xi^T \end{bmatrix}. & (5.21) \end{aligned}$$

It is possible to introduce one more shrewdness. First, it has to be noticed that the function $g(\cdot)$ that adds a new landmark is usually independent by the existing landmarks. Secondly, a new landmark is perceived by sensor measurement ($\mathbf{s}$), thus we have to compose the measurement with the current robot position ($\boldsymbol{\Gamma}_k$) in order to generate its position in world coordinates and to add it to the current map. This implies that the current map landmarks are not involved in the addition of a new landmark and the Jacobians of Equation 5.19

are sparse even more. As in the prediction step, we can split the state vector (before the landmark addition) in the dynamic part $\mathbf{D}$, the map part $\mathbf{m}$. Then the Jacobian of $g(\cdots)$ with respect to the state vector results

$$\mathbf{G}_\mathbf{x} \;=\; \left[\left.\frac{\partial g(\cdots)}{\partial \mathbf{D}}\right|_{\mathbf{D}=\bar{D}_k,\mathbf{s}=\bar{\mathbf{s}},\boldsymbol{\xi}=\mathbf{0}} \quad \left.\frac{\partial g(\cdots)}{\partial \mathbf{m}}\right|_{\mathbf{D}=\bar{D}_k,\mathbf{s}=\bar{\mathbf{s}},\boldsymbol{\xi}=\mathbf{0}} \right] = \begin{bmatrix}\mathbf{G}_\mathbf{D} & \mathbf{0}\end{bmatrix}. \tag{5.22}$$

Consequently, the covariance matrix computation can be simplified even more:

$$\boldsymbol{\Sigma}_k^* \;=\; \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I} \\ \mathbf{G}_\mathbf{D} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{DD}k} & \boldsymbol{\Sigma}_{\mathbf{Dm}k} \\ \boldsymbol{\Sigma}_{\mathbf{Dm}k}^T & \boldsymbol{\Sigma}_{\mathbf{mm}k} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 & \mathbf{G}_\mathbf{D}^T \\ 0 & \mathbf{I} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{G}_\xi \boldsymbol{\Sigma}_\xi \mathbf{G}_\xi^T \end{bmatrix}^T \tag{5.23}$$

$$= \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{DD}k} & \boldsymbol{\Sigma}_{\mathbf{Dm}k} & \boldsymbol{\Sigma}_{\mathbf{DD}k}\mathbf{G}_\mathbf{D}^T \\ \boldsymbol{\Sigma}_{\mathbf{Dm}k}^T & \boldsymbol{\Sigma}_{\mathbf{mm}k} & \boldsymbol{\Sigma}_{\mathbf{Dm}k}^T\mathbf{G}_\mathbf{D}^T \\ \mathbf{G}_\mathbf{D}\boldsymbol{\Sigma}_{\mathbf{DD}k} & \mathbf{G}_\mathbf{D}\boldsymbol{\Sigma}_{\mathbf{Dm}k} & \mathbf{G}_\mathbf{D}\boldsymbol{\Sigma}_{\mathbf{DD}k}\mathbf{G}_\mathbf{D}^T + \mathbf{G}_\xi\boldsymbol{\Sigma}_\xi\mathbf{G}_\xi^T \end{bmatrix}. \tag{5.24}$$

Although at first glance the latter equation seems to be more complex than the Equation 5.21, it has to be noticed that $\mathbf{G}_\mathbf{D}$ is a matrix with the number or rows equal to the dimension of the new landmark vector and columns equal to the dimension of the dynamic part, while $\mathbf{G}_\mathbf{x}$ has the number of columns equal to the entire state vector dimension. Thanks to this, the landmark addition step has a computational complexity of $O(n)$, being n the number of landmarks in the map.

5.4.5 Measurement

The update step of the standard EKF can be revisited in two different steps in the realm of EKF-SLAM: the *measurement step* and the *update step*.

In a SLAM system, we are assuming that sensors perceive some kind of measure from surrounding landmarks, but not necessarily for all the landmarks. In fact, real sensors have operative limits (e.g., maximum and/or minimum distance) and some landmarks that are in the map cannot be measured, or some landmarks can be temporarily occluded. The measurement step (see steps 9-15 of Algorithm 5) allows, for each landmark $\mathbf{y}_i$, to predict the expected measure according to a measurement model $h_i(\mathbf{x}, \mathbf{v}, \boldsymbol{\delta}_i)$, being $\boldsymbol{\delta}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\delta}i})$ the associated noise model and $\mathbf{v}$ some external input related to the specific measure.

It is possible to calculate the value

$$\mathbf{h}_i \;=\; h_i(\boldsymbol{\mu}_k, \mathbf{v}_i, \boldsymbol{\delta}_i = 0), \tag{5.25}$$

for each landmark i in the map, representing the expected value of the measure (or measurement prediction). The covariance of the measure can be retrieved by the computation of the matrix

$$\mathbf{S}_i \;=\; \mathbf{H}_\mathbf{x}\boldsymbol{\Sigma}_k\mathbf{H}_\mathbf{x}^T + \mathbf{H}_\delta\boldsymbol{\Sigma}_{\boldsymbol{\delta}i}\mathbf{H}_{\boldsymbol{\delta}}^T, \tag{5.26}$$

where

$$\mathbf{H}_{i\mathbf{x}} \;=\; \left.\frac{\partial h_i(\cdots)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu},\mathbf{v}=\mathbf{v}_i,\boldsymbol{\delta}_i=0}, \tag{5.27}$$

and

$$\mathbf{H}_{i\boldsymbol{\delta}} = \left.\frac{\partial h_i(\cdots)}{\partial \boldsymbol{\delta}}\right|_{\mathbf{x}=\boldsymbol{\mu},\mathbf{v}=\mathbf{v}_i,\boldsymbol{\delta}_i=0}, \tag{5.28}$$

are the Jacobians of the measurement function with respect to the state vector and the measurement noise model, evaluated at the current estimation of the vector state.

It is noticeable that Jacobian $\mathbf{H}_{i\mathbf{x}}$ is a sparse matrix in SLAM. As a matter of fact, the measurement function $h_i(\cdots)$ is as a transformation composition that involves a single landmark, expressed w.r.t. the world reference frame, and the current robot position w.r.t the robot reference frame. Therefore it can be reformulated as

$$h_i(\boldsymbol{\Gamma}, \mathbf{y}, \mathbf{v}, \boldsymbol{\delta}_i), \tag{5.29}$$

and the Jacobian matrix shows sparsity

$$\mathbf{H}_{i\mathbf{x}} = \begin{bmatrix}\mathbf{H}_{i\boldsymbol{\Gamma}} & 0 & \dots & 0 & \mathbf{H}_{i\mathbf{y}_i} & 0 & \dots & 0\end{bmatrix}, \tag{5.30}$$

being

$$\mathbf{H}_{i\boldsymbol{\Gamma}} = \left.\frac{\partial h_i(\cdots)}{\partial \boldsymbol{\Gamma}}\right|_{\boldsymbol{\Gamma}=\bar{\boldsymbol{\Gamma}},\mathbf{y}_i=\bar{\mathbf{y}}_i\mathbf{v}=\mathbf{v}_i,\boldsymbol{\delta}_i=0}, \tag{5.31}$$

$$\mathbf{H}_{i\mathbf{y}_i} = \left.\frac{\partial h_i(\cdots)}{\partial \mathbf{y}}\right|_{\boldsymbol{\Gamma}=\bar{\boldsymbol{\Gamma}},\mathbf{y}_i=\bar{\mathbf{y}}_i\mathbf{v}=\mathbf{v}_i,\boldsymbol{\delta}_i=0}, \tag{5.32}$$

where the non zero block are the first (corresponding to the $\boldsymbol{\Gamma}$ element in the state vector) and the i-th (corresponding to the measured landmark). The computation of each measurement prediction has constant complexity, consequently the measurement prediction step has linear complexity in the number of landmarks. The computation of the measure covariance can be skipped for the landmarks for which the expected value of the measurement reveals that the sensors will not be able perceive them (e.g., points which are predicted to lie behind a camera), reducing the complexity of the measurement step.

Each measurement prediction $\mathbf{h}_i$, together with its associated covariance $\mathbf{S}_i$, is a useful information that allows an *individual compatibility test* by the evaluation of the *innovation*, expressed by the *squared Mahalanobis Distance*

$$d = (\mathbf{z}_i - \mathbf{h}_i)^T \mathbf{S}_i^{-1} (\mathbf{z}_i - \mathbf{h}_i). \tag{5.33}$$

The value assumed by d allows the selection of a set of landmarks which measure $\mathbf{z}_i$ is in the “expected range”. Let consider $M = \{m_1, m_2, \dots, m_m\} \subset \{1, 2, \dots, n\}$ the set of the index of the landmarks which expected measure is individually compatible with the predicted measure, where a measure is said to be individually compatible if

$$d < \mathbf{F}^{-1}_{\chi^2_r}(\alpha), \tag{5.34}$$

being $\mathbf{F}^{-1}_{\chi^2_r}$ the inverse of the cumulative distribution function of a χ^2 variable with r degrees of freedom, where r is the dimension of the measurement vector $\mathbf{h}_i$, evaluated at α percentile. Common values of α are between 0.95 and 0.99.

This simple test allows to identify some possible outliers, i.e., measurements that assume a very low probable value and thus are probably due to some unmodeled error, e.g., a wrong data association. Moreover, the prediction on the measurement value with its associated covariance allows in some applications, an *active search approach*, i.e., it is possible to "ask" the sensor to perform a measure in the predicted range, saving time and reducing the chance of data association errors. This test does not provide a certain data association mechanism, thus outliers exists and more complex rejection methods needs to be considered.

A complete prediction of measurements is obtained by iteratively append the expected measure $\mathbf{h}_{m_i}$ with the acquired $\mathbf{z}_{m_i}$ sensor measure for all the individual compatibles measurements, i.e., for $m_i \in M$:

$$\begin{aligned} \mathbf{h} &\leftarrow [\mathbf{h};\ \mathbf{h}_{m_i}]; && (5.35)\\ \mathbf{z} &\leftarrow [\mathbf{z};\ \mathbf{z}_{m_i}]; && (5.36)\\ & && (5.37) \end{aligned}$$

Jacobian matrices and noise model covariances are properly stacked to $\mathbf{H}$ and to the block diagonal matrix $\mathbf{R}$:

$$\begin{aligned} \mathbf{H} &= [\mathbf{H};\ \mathbf{H}_{m_i\mathbf{x}}], && (5.38)\\ \mathbf{R} &= diag(\mathbf{R},\ \mathbf{H}_{m_i\nu}\mathbf{\Sigma}_\nu\mathbf{H}_{m_i\nu}^T). && (5.39) \end{aligned}$$

It is noticeable that the $\mathbf{H}$ matrix is sparse, being composed by the sparse matrix of Equation 5.30

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_{m_1}(\cdots)}{\partial \mathbf{\Gamma}} & \frac{\partial h_{m_1}(\cdots)}{\partial \mathbf{y}_{m_1}} & 0 & 0 & 0 \\ \frac{\partial h_{m_2}(\cdots)}{\partial \mathbf{\Gamma}} & 0 & \frac{\partial h_{m_2}(\cdots)}{\partial \mathbf{y}_{m_2}} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_{m_i}(\cdots)}{\partial \mathbf{\Gamma}} & 0 & 0 & \frac{\partial h_{m_i}(\cdots)}{\partial \mathbf{y}_{m_i}} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_{m_m}(\cdots)}{\partial \mathbf{\Gamma}} & 0 & 0 & 0 & \frac{\partial h_{m_m}(\cdots)}{\partial \mathbf{y}_{m_m}} \end{bmatrix} \qquad (5.40)$$

5.4.6 Update Step

In the measurement step, a set of observations was created and the measurements are stored in vectors $\mathbf{h}$ and $\mathbf{z}$ and matrices $\mathbf{H}$ and $\mathbf{R}$. The Kalman gain is then computed as

$$\begin{aligned} \mathbf{S} &= \mathbf{H}\hat{\mathbf{\Sigma}}_k\mathbf{H} + \mathbf{R}, && (5.41)\\ \mathbf{K} &= \hat{\mathbf{\Sigma}}_k\mathbf{H}^T\mathbf{S}^{-1}, && (5.42) \end{aligned}$$

and the filter state is updated accordingly to Equation 4.88 and Equation 4.89. Notice that, in case there are no measurements available, the update step is skipped and the prediction state is assumed as the new state. The complexity of this step is $O(mn^2 + m^2n)$, being m the number of measured landmark and n the total number of landmarks. In practical cases the number of landmark measured per step is significantly lower than the number of landmarks in the map, thus the complexity of the update step results $O(n^2)$.

5.4.6.1 Iterated Updates

An alternative way to perform the EKF update step reckon on an iterated update step performed in the following way:

1. stack the measurements in $\mathbf{h}$ and $\mathbf{z}$ vector in decreasing order of *innovation*, evaluated with Equation 5.33
2. integrate one measure at the time recomputing its measurement prediction and the Jacobians at each step.

This procedure provides a better linearization of the Jacobians, since they are recomputed on the updated state, but it adds complexity to the update steps. Moreover, this procedure is prone to outliers, since it integrates at first the most innovative measurements (having high probability of being outliers) with the effect of a big “change” in the state values. Practically, this kind of updates is suitable in simulation, where the data association is given and there are no outliers.

5.4.6.2 1-Point RANSAC Update

In practical cases, the method presented up to here for the update of the filter may fail as well in the presence of outliers, e.g., wrong data associations. With real data, the update step has to be complemented by a robust data association procedure. Different techniques have been developed to this extend, in particular one of the simplest and most effective is the 1-Point RANSAC (RANdom SAmpling and Consensous) technique presented in [16] [15] that executes two update steps on two disjoint sets of measurements characterized, respectively, by low and high innovation discarding possible outliers. An alternative technique is named *Joint Compatibility Branch and Bound (JCBB)* [68], but it is not considered in this work.

A single iteration of the 1-Point RANSAC update step selects randomly a measurement prediction $\mathbf{h}_R$ from the individually compatible measurements and integrates it performing an update step of the mean vector only, i.e., it use Equation 4.88 only. Then, the entire set of measurements $\{\mathbf{h}_M\}$ is recomputed (the computation of covariances matrix $\mathbf{S}_i$ is not necessary) and the measurements that are close to the corresponding observations $\mathbf{z}_i$ are added to the set $\{\mathbf{h}_{LI}\}$, i.e., the vector of the measurements with low innovation (LI). The distance between a measurement prediction and its correspondent observation is evaluated by the standard Euclidean distance. Then, the mean vector of the EKF state previous to the update step is restored and the random selection is repeated (i.e., a new set $\{\mathbf{h}_{LI}\}$ is generated) until a termination criterion on the number of iteration has been reached. The set $\{\mathbf{h}_{LI*}\}$ with the highest number of elements is stored during the repetition of the random selection and it is used to perform a complete update step (i.e., using Equation 4.88 and Equation 4.89), named *low innovation update step*.

After the low innovation update step, the remaining measurements predictions $\{\mathbf{h}_j | \mathbf{h}_j \notin \mathbf{h}_{LI^*}\}$, i.e., those that are not in the $\{\mathbf{h}_{LI^*}\}$ set, are reevaluated together with their covariances $\mathbf{S}_j$. The individual compatibility, evaluated with Equation 5.33, of each measurement prediction $\mathbf{h}_j$ with the observation $\mathbf{z}_j$ is checked and compatible measurements are stored in the high innovation measurement set $\{\mathbf{h}_{HI}\}$ and used in a second update step, the *high innovation update step*. The remaining measurements, i.e., the onces that are not individually compatible, are discarded as outliers.

The RANSAC selection terminates after n_{it} iterations has been performed. The number of iteration n_{it} is initialized with a given value n_0 ant it is adaptively computed as

$$n_{it} = \frac{\log(1-p)}{\log \epsilon}, \tag{5.43}$$

where

$$\epsilon = \frac{\#\{\mathbf{h}_M\} - \#\{\mathbf{h}_{LI^*}\}}{\#\{\mathbf{h}_M\}}. \tag{5.44}$$

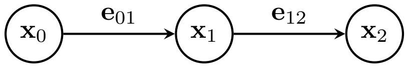
The Equation 5.43 compute the number of iterations, i.e., the number of tested hypothesis, needed to ensure that at least one set of measurements without outliers has been tested with an high probability p. Typical values for p are 0.95 or 0.99. Equation 5.44 estimates ϵ, the ratio between the number of measures that are not selected as candidates for the low innovation step and the total number of measurements. The complete update step performed with the 1-Point RANSAC technique is summarized in Algorithm 6.

Algorithm 6 onePointRansac($\{\mathbf{h}_i\}$, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$)

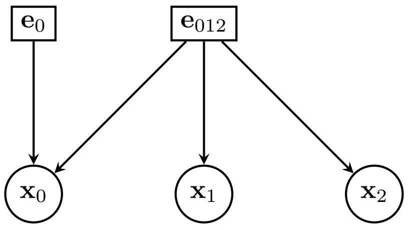
```
1:  $\boldsymbol{\mu}_I \leftarrow \boldsymbol{\mu}$, $it \leftarrow 0, n_{it} \leftarrow n_0, \{\mathbf{h}_{LI*}\} \leftarrow \emptyset$
2:  # ***RANSAC selection loop***
3:  **while** $it < n_{it}$ **do**
4:      select $\mathbf{h}_R$ randomly from $\{\mathbf{h}_M\}$
5:      $\boldsymbol{\mu} \leftarrow$ EKF update step with $\mathbf{h}_R, \mathbf{z}_R$
6:      $\{\mathbf{h}_{LI}\} \leftarrow \{\mathbf{h}_i | \mathbf{h}_i \in \{\mathbf{h}_M\}, \mathrm{dist}(\mathbf{h}_i, \mathbf{z}_i) < d_{LI}\}$
7:      **if** $\#\{\mathbf{h}_{LI}\} > \{\#\mathbf{h}_{LI*}\}$ **then**
8:          $\{\mathbf{h}_{LI*}\} \leftarrow \{\mathbf{h}_{LI}\}$
9:      **end if**
10:     $\epsilon \leftarrow \frac{\#\{\mathbf{h}_M\} - \#\{\mathbf{h}_{LI}\}}{\#\{\mathbf{h}_M\}}$
11:     $n_{it} \leftarrow \frac{\log(1-p)}{\log \epsilon}$
12:     $it \leftarrow it + 1$
13:     $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_I$
14: **end while**
15: # ***Low innovation update step***
16: $\boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ EKF update step with $\{\mathbf{h}_{LI*}\}, \{\mathbf{z}_{LI*}\}$
17: # ***High innovation selection, outliers rejection***
18: recompute measures prediction of $\{\mathbf{h}_M \backslash \mathbf{h}_{LI*}\}$
19: $\{\mathbf{h}_{HI}\} \leftarrow \{\mathbf{h}_M \backslash \mathbf{h}_{LI*}|$ individually compatible with $\mathbf{z}_i\}$
20: $\boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ EKF update step with $\{\mathbf{h}_{HI}\}, \{\mathbf{z}_{HI}\}$
```

5.5 Graph Formulation

So far we have focused on the online SLAM problem and how it has been addressed through the use of a EKF. Here we introduce the full SLAM problem (i.e., the estimation of the posterior probability $\boldsymbol{p}(\boldsymbol{\Gamma}_{1:T}, \mathbf{m}|\mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \boldsymbol{\Gamma}_0)$) and how it can be formulated as a problem of optimization on a graph (different from the DBN graph), where nodes represent the variables to optimize and edges represent constraints between nodes.



(a) Graph example with binary edges



(b) Graph example with ternary edges

Figure 5.5: *Some graph example. (a) presents three nodes and two binary edges, (b) presents three nodes with a ternary edge and a unary edge.*

5.5.1 Optimization On a Graph

Let consider $\mathbf{x}$, a vector of parameters that has to be estimated, partitioned as:

$$\mathbf{x} \;=\; \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \dots & \mathbf{x}_{N-1}^T & \mathbf{x}_N^T \end{bmatrix}^T. \tag{5.45}$$

where each $\mathbf{x}_i, i = 1 : N$ is itself a vector and it will be represented by a *node* in a graph $\mathcal{G}$. Let $\mathbf{z}_{ij}$ and $\mathbf{\Omega}_{ij}$ be respectively the mean and the information matrix (i.e., the inverse of a covariance matrix) of a "virtual measurement" between nodes i and j. A virtual measurement can be tough as function

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}), \tag{5.46}$$

that is zero when the parameters satisfy fully the expected value $\mathbf{z}_{ij}$. Each virtual measure is represented by an *edge* in the set of the edges $\mathcal{E}$ that links nodes in the graph. To compact the notation, the edge is usually identified by $\mathbf{e}_{ij}$, leaving implicit the parameters. The presented virtual measurement is represented by a *binary* edge, since it involves two nodes. Being more general, an edge can involve a generic number of nodes: for instance, it can be an *unary* edge $\mathbf{e}_i(\mathbf{x}_i, \mathbf{z}_i) = \mathbf{e}_i$ with information matrix $\mathbf{\Omega}_i$ or a *ternary* edge $\mathbf{e}_{ijk}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{z}_{ijk}) = \mathbf{e}_{ijk}$ accompanied by information matrix $\mathbf{\Omega}_{ijk}$. Figure 5.5(a) shows a graph with three nodes ($\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$) and two binary edges ($\mathbf{e}_{01}$, $\mathbf{e}_{12}$), while Figure 5.5(b) shows a graph with three nodes ($\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$), a ternary edge $\mathbf{e}_{012}$ connecting them and a unary edge $\mathbf{e}_0$ that represent a constraint on the node $\mathbf{x}_0$. Notice that the edge representation is enhanced in some cases with a square node that clarifies, especially for non binary edges, the nodes involved in a constraint. Moreover, edges are not directed, but in graphical representation we prefer to use directed edges to clarify the role of the nodes involved in the constraint. Let consider, for instance, a binary edge between two nodes that represents two successive values of a variable in time. It is intuitive to represent the edge with an arrow going from the oldest node to the new one. When non binary edges are considered, arrows exit from the square node representing the edge and enter in the nodes involved in the constraint.

Considering the set $\mathcal{E}$ of edges and their respective information matrices of a graph, the goal of a minimization algorithm is to find the parameters $\mathbf{x}^*$ that minimize the function

$$\mathbf{F}(\mathbf{x}) \;=\; \sum_{\mathbf{e}, \mathbf{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{e}(\cdots)^T \, \mathbf{\Omega}_\mathbf{e} \, \mathbf{e}(\cdots), \tag{5.47}$$

where the parameters of the function $\mathbf{e}$ are leaved unspecified since they depends on the single edge structure. The minimization can be performed through one of the *Non Linear Least Square* minimization algorithm presented in Section 4.3 after a rewriting of the function $\mathbf{F}$. First of all, each edge can be expressed as function of the entire parameters vector $\mathbf{x}$, instead of its own nodes. For instance, a binary edge $\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ can be equivalently expressed as $\mathbf{e}_{ij}(\mathbf{x})$, where to compact the notation the $\mathbf{z}_{ij}$ element is skipped. The goal function can be rewritten as

$$\begin{aligned} \mathbf{F}(\mathbf{x}) &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{e}(\mathbf{x})^T \boldsymbol{\Omega}_\mathbf{e} \, \mathbf{e}(\mathbf{x}) && (5.48) \\ &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{e}(\mathbf{x} \boxplus \Delta\mathbf{x})^T \boldsymbol{\Omega}_\mathbf{e} \, \mathbf{e}(\mathbf{x} \boxplus \Delta\mathbf{x}), && (5.49) \end{aligned}$$

and considering the first order Taylor expansion

$$\begin{aligned} \mathbf{F}(\mathbf{x}) &\simeq \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} (\mathbf{e}(\mathbf{x}) + \mathbf{J}_\mathbf{e}\Delta\mathbf{x})^T \Omega_\mathbf{e}(\mathbf{e}(\mathbf{x}) + \mathbf{J}_\mathbf{e}\Delta\mathbf{x}) && (5.50) \\ &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{e}(\mathbf{x})^T \Omega_\mathbf{e}\mathbf{e}(\mathbf{x}) + 2\mathbf{e}(\mathbf{x})^T \Omega_\mathbf{e}\mathbf{J}_\mathbf{e}\Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{J}_\mathbf{e}^T \Omega_\mathbf{e}\mathbf{J}_\mathbf{e}\Delta\mathbf{x} && (5.51) \\ &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} c_\mathbf{e} + 2\mathbf{b}_\mathbf{e}^T \Delta\mathbf{x} + \mathbf{x}^T \mathbf{H}_\mathbf{e} \, \mathbf{x}, && (5.52) \end{aligned}$$

the terms

$$\begin{aligned} c &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} c_\mathbf{e}, && (5.53) \\ \mathbf{b} &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{b}_\mathbf{e}, && (5.54) \\ \mathbf{H} &= \sum_{\mathbf{e},\boldsymbol{\Omega}_\mathbf{e} \in \mathcal{E}} \mathbf{H}_\mathbf{e}, && (5.55) \end{aligned}$$

shows that this function is equivalent to the one presented in Equation 4.99, thus the system can be solved by the algorithms presented in Section 4.3.

It is important to say that the matrix $\mathbf{H}$ is sparse by construction. The Jacobian matrix $\mathbf{J}_\mathbf{e}$ of a single edge $\mathbf{e}$

$$\mathbf{J}_\mathbf{e} = \left. \frac{\partial \mathbf{e}(\tilde{\mathbf{x}} \boxplus \Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \right|_{\Delta\mathbf{x}=\mathbf{0}, \tilde{\mathbf{x}}=\mathbf{x}}, \qquad (5.56)$$

is sparse, because the edge $\mathbf{e}$ involves only part of the parameters $\mathbf{x}$, i.e., the edge connects a small number of nodes of the graph. For instance, considering a binary edges $\mathbf{e}_{ij}$ involving nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ with $i \neq j$, the Jacobian has the following structure

$$\mathbf{J}_\mathbf{e} = \begin{bmatrix} 0 \cdots 0 & \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}{\partial \mathbf{x}_i} & 0 \cdots 0 & \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}{\partial \mathbf{x}_j} & 0 \cdots 0 \end{bmatrix}, \qquad (5.57)$$

where the non zero blocks are in the block column i and j. This implies that $\mathbf{J}_\mathbf{e}$ has a number of non zero blocks equal to the number of nodes involved in the edge. As

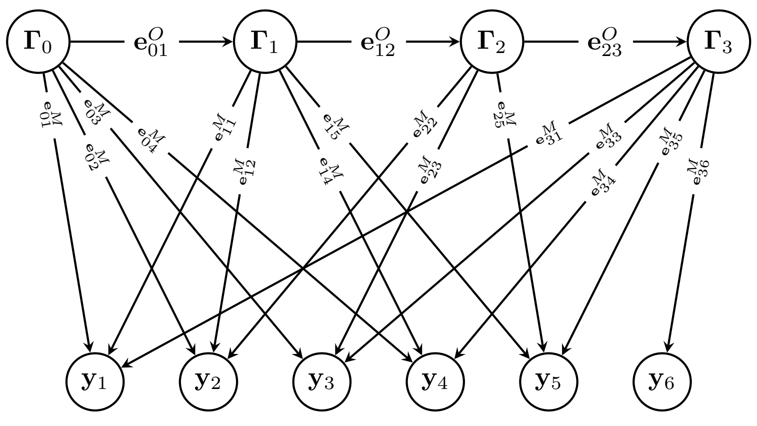


Figure 5.6: *An example of a SLAM problem represented on a graph.* $\mathbf{\Gamma}$ *nodes represent robot poses,* $\mathbf{y}$ *nodes represent landmarks,* $\mathbf{e}^O$ *edges represent relative displacement of robot poses,* $\mathbf{e}^M$ *edges represent the measurement of the landmarks. It can be noticed that the problem is exactly the same of the one represented in Figure 5.4.*

a consequence, the $\mathbf{H}$ matrix is sparse, since it comes from the summation of the $\mathbf{H}_\mathbf{e}$ matrices, given by $\mathbf{H}_\mathbf{e} = \mathbf{J}_\mathbf{e}^T \mathbf{\Omega}_\mathbf{e} \mathbf{J}_\mathbf{e}$. The solution of the system Equation 4.101 can be computed in a efficient way thanks to sparse Cholesky factorization algorithm. A very useful tool, freely available with libraries and examples, for the graph based optimization is the g^{2}o framework presented in [49].

5.5.2 SLAM On a Graph

A SLAM problem can be formulated with the support of a graph formalism, as stated in [35], considering two types of nodes: the robot pose nodes $\mathbf{\Gamma}_i$ and the landmark nodes $\mathbf{y}_i$. The edges are of two different types: the first connect pose nodes and impose a constraint on the relative displacement between two consecutive poses, e.g., the odometric measure of robot displacement; the second type connect a pose node and a landmark node, being a constraint on the measurement of the landmark gathered when the robot was in a specific position. The edges are named respectively $\mathbf{e}^O(\mathbf{\Gamma}_i, \mathbf{\Gamma}_j, \mathbf{z}_{ij})$ and $\mathbf{e}^M(\mathbf{\Gamma}_i, \mathbf{y}_j, \mathbf{z}_{ij})$, representing respectively the constraint due to the odometric data and the constraint due to the measure of the environment. Edges $\mathbf{e}^O$ can be omitted in case the information of the displacement are not available (e.g., if the odometric information are not gathered by the robot). An example of a graph representing a small SLAM problem is depicted in Figure 5.6. It can be noticed that it represent exactly the same example of the DBN of Figure 5.4.

This kind of optimization is known also as *Bundle Adjustment* and its solution was widely studied over years in the Structure From Motion research field, as stated in the survey presented in [104]. In particular, it has been proven that a proper ordering of the edges allows to treat the problem in a very efficient way. Let consider the nodes ordered such that robot pose nodes have the lower indexes in the vector of parameters and landmarks are in the last part. The $\mathbf{H}$ and the vector $\mathbf{b}$ can be partitioned and the system of

Equation 4.101 can be reduced to

$$\begin{bmatrix} \mathbf{H}_{\mathbf{\Gamma\Gamma}} & \mathbf{H}_{\mathbf{\Gamma y}} \\ \mathbf{H}_{\mathbf{\Gamma y}}^T & \mathbf{H}_{\mathbf{yy}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_{\mathbf{\Gamma}} \\ \Delta\mathbf{x}_{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} -\mathbf{b}_{\mathbf{\Gamma}} \\ -\mathbf{b}_{\mathbf{y}} \end{bmatrix} \tag{5.58}$$

where the subscript $\mathbf{\Gamma}$ indicates the robot pose nodes and $\mathbf{y}$ indicates the landmarks nodes. Thanks to the Shur complement, the solution can be computed in two steps, the first retrieves the solution for $\Delta\mathbf{x}_{\mathbf{\Gamma}}^*$ from

$$\left(\mathbf{H}_{\mathbf{\Gamma\Gamma}} - \mathbf{H}_{\mathbf{\Gamma y}}\mathbf{H}_{\mathbf{yy}}^{-1}\mathbf{H}_{\mathbf{\Gamma y}}^T\right)\Delta\mathbf{x}_{\mathbf{\Gamma}} = -\mathbf{b}_{\mathbf{\Gamma}} + \mathbf{H}_{\mathbf{\Gamma y}}\mathbf{H}_{\mathbf{yy}}^{-1}\mathbf{b}_{\mathbf{y}}, \tag{5.59}$$

and then the solution $\Delta\mathbf{x}_{\mathbf{y}}^*$ for

$$\mathbf{H}_{\mathbf{\Gamma\Gamma}}\Delta\mathbf{x}_{\mathbf{y}} = -\mathbf{b}_{\mathbf{y}} - \mathbf{H}_{\mathbf{\Gamma y}}^T\Delta\mathbf{x}_{\mathbf{\Gamma}}^*. \tag{5.60}$$

If no odometric edges are used, the $\mathbf{H}_{\mathbf{\Gamma\Gamma}}$ is block diagonal, since there exists no edge that involves more than one camera. If the odometric edges are used, $\mathbf{H}_{\mathbf{\Gamma\Gamma}}$ is block tri-diagonal. In both cases, $\mathbf{H}_{\mathbf{\Gamma\Gamma}}$ can be easily inverted. Moreover, in a typical Bundle Adjustment problem, the number of landmarks is much more larger than the camera poses, thus the solution of Equation 5.59 can be solved faster despite the additional time spent in calculating the left side term of the equation.

5.5.3 Gauge of Freedom

The optimization algorithm can be properly executed only if the Hessian matrix $\mathbf{H}$ of Equation 5.55 is invertible, i.e., if it is not singular. In Section 4.3.1 we have explored one of the conditions that can causes singularity in the Hessian matrix, i.e., the presence of over parametrized representations of the parameters. This problem has been addressed by the redefinition of the composition operator for the increments that works in a local space with a minimal representation. A second source of singularity can originate from the problem description itself and this is the case of the SLAM problem. The solution of the off-line SLAM problem is constituted by the complete robot trajectory and the map, i.e., the landmark locations. The trajectory and the landmarks can be thought as a rigid body, consequently a rigid transformation of the solution does not affects the error value $\mathbf{F}(x)$ of Equation 5.47, i.e., the solution is under determined. This characteristic is reflected by the singularity of the Hessian matrix and more precisely the Hessian matrix is rank deficient of 6 degrees, i.e., the solution has 6 degrees of freedom and it can not be determined.[2]

To cope with this problem we have to *fix* a proper number of elements of the parameters vector, i.e., to exclude them from the optimization by consider their values known. In the SLAM case it is sufficient to fix the elements corresponding to a robot pose node, which fixes exactly 6 degrees of freedom. Without loss of generality, the first node can be considered fixed, i.e., the starting pose of the SLAM trajectory and map can be considered known and fixed, for instance, in the origin. The fixed nodes have to be excluded from the estimation process, i.e., they do not have to appear in the parameters vector and in the Hessian matrix. Consequently when the Jacobian of a generic edge is computed, the computation of the Jacobians of this edge with respect to the fixed nodes have to be skipped.

[2]It can be noticed that the solution can not be determined with the Gauss-Newton algorithm, which requires a full rank Hessian matrix, while the Lenveberg-Marquadt algorithm solves under determined problems thanks to the use of the variable damping factor λ.

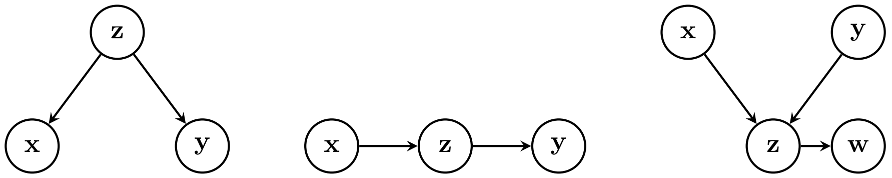
5.6 Conditionally Independent Sub-Maps SLAM

The EKF-SLAM algorithm suffers from some limitations. First, the complexity of the algorithm is dominated by the update step, which is $O(n^2)$ where n is the number of landmarks in the map. This implies that when the number of landmarks grows, i.e., the environment to explore is not limited to an experimental small setup, EKF-SLAM looses quickly the ability to run in real time. Secondly, EKF formulation uses a linear approximations for the Jacobians and this produce optimistic estimation for the covariance matrix that usually results in an inconsistent behavior of the filter.

Sub-Mapping techniques comes the overcome both limitations by splitting the explored area into several subproblems, each of them with a bounded number of landmarks. This implies that the computational complexity is bounded and can be considered constant for each submap. The *Conditional Independence Sub-Maps* (CI) framework, presented in [78] and [77], represents one of the smartest and effective sub-mapping technique. Here the so called *local* submap version is presented, while the *global* submap technique is treated briefly only to explain some basic concepts. The term *local* indicates that each submap of the problem store landmarks in a local reference frame, while the transformation that links the local reference frame with the global one is stored apart. On the contrary, the *global* submap approach simply split the entire map in subproblems, but the landmarks of the maps are stored with respect to an unique global reference system. The advantages of the local submaps approach, although there is no a formal proof, come from the evidence that local coordinates approaches bound the uncertainty, reducing the approximation introduced by linearizations. The convenience of the CI slam technique lies in the ability to perform large scale SLAM, i.e., to operate on real dataset, without introducing any approximations besides the inherent EKF linearizations. The CI submaps algorithm is briefly summarized in the following paragraphs, while details are given in the next sections.

Each submap can be treated as an EKF-SLAM algorithm, where as usual the robot starts in the origin and with no uncertainty, but in general the map is not empty, i.e. some landmarks are already in the map coming from the previous local submap. When a sufficient number of landmarks is in the current map, or the map has covered an adequate area, the map is “closed” and a new map started. The pose of the robot in the closed map represents the starting pose of the new map, thus, to know the global coordinates of a map it is necessary to compose all the previous relative transformation between maps. The landmarks in the current map that come from the previous map (let call them the *shared landmarks*) have an initial estimate, given by the estimation performed in the previous map.

The estimation performed in the new map contributes to the refinement of the estimation of the shared landmarks and the new information can be *back-propagated* to the previous maps to improve the global estimation in linear time in terms of number of maps. A special procedure allow to treat *loop closures*, i.e., the observation of landmarks that are in a previous maps. Thanks to a different dedicated procedure, sub maps can be *reused* when it is useful, i.e., when the robot traverse an area that was already explored. This last procedure is not explicitly treated and it is not used in this work but it is conceptually similar to the treated ones; refer to the original work [76] for further details.



(a) "tail-to-tail", x and y conditionally independent given z

(b) "head-to-tail", x and y conditionally independent given z

(c) "head-to-head", x and y independent but not conditionally independent given z or a descendant node

Figure 5.7: *Dynamic Bayesian Network (DBN) of three variables showing conditional independence and independence properties.*

5.6.1 From DBN to CI submap

The Conditionally Independent Sub-Maps SLAM framework is based on the concept of Conditional Independence between stochastic variables. Although this concept has been presented in the Section 4.1.6, we have to introduce it again, highlighting its relationship with the Dynamic Bayesian Network formalism. In this section we explain the theoretical concepts that bring to the comprehension of the CI SLAM framework underlying mechanism, while in next sections we will give more practical details on its practical implementation.

Figure 5.7 shows three simple DBN. Each of them will be considered in the following in order to explain and to put in evidence the conditional independence concept, that can be retrieved in a DBN where the d-separation property exists. This explanation is strongly inspired from what done in [5], that can be considered by the interested reader for further details.

5.6.1.1 Independence and Conditional Independence on DBN

Lets consider the simple Bayesian network of Figure 5.7(a), representing the joint probability $\boldsymbol{p}(x, y, z)$ of the three variable x, y and z. Following the edges it is possible to express the joint probability as

$$\boldsymbol{p}(x, y, z) \quad = \quad \boldsymbol{p}(x|z)\boldsymbol{p}(y|z)\boldsymbol{p}(z). \tag{5.61}$$

By marginalizing z it can be proven that x and y are not independent

$$\boldsymbol{p}(x, y) \quad = \quad \int_z \boldsymbol{p}(x|z)\boldsymbol{p}(y|z)\boldsymbol{p}(z) \quad \neq \quad \boldsymbol{p}(x)\boldsymbol{p}(y). \tag{5.62}$$

Now consider the conditional probability of x and y given z. By definition it can be expressed by the following ratio

$$\boldsymbol{p}(x, y|z) \quad = \quad \frac{\boldsymbol{p}(x, y, z)}{\boldsymbol{p}(z)} \quad = \quad \boldsymbol{p}(x|z)\boldsymbol{p}(y|z), \tag{5.63}$$

that implies that x and y are conditionally independent given z as stated in Equation 4.34. Graphically it is possible to notice that the node z is *tail-to-tail* with respect to x and y, i.e., it is connected by the tail of the arrows to the nodes x and y, and it acts as a "block" on the path between x and y.

In Figure 5.7(b) the joint probability $\boldsymbol{p}(x, y, z)$ can be expressed as

$$\boldsymbol{p}(x, y, z) = \boldsymbol{p}(z|x)\boldsymbol{p}(y|z)\boldsymbol{p}(x), \tag{5.64}$$

By marginalizing z it can be proven that x and y are not independent

$$\boldsymbol{p}(x, y) = \int_z \boldsymbol{p}(z|x)\boldsymbol{p}(y|z)\boldsymbol{p}(x) \neq \boldsymbol{p}(x)\boldsymbol{p}(y). \tag{5.65}$$

Now consider the conditioned probability of x and y given z. By definition it can be expressed by the following ratio

$$\boldsymbol{p}(x, y|z) = \frac{\boldsymbol{p}(x, y, z)}{\boldsymbol{p}(z)} = \boldsymbol{p}(y|z)\frac{\boldsymbol{p}(z|x)\boldsymbol{p}(x)}{\boldsymbol{p}(z)} = \boldsymbol{p}(y|z)\boldsymbol{p}(x|z), \tag{5.66}$$

and, thanks to the Bayes rule, it shows conditional independence of x and y given z. The node z is said to be "head-to-tail" with respect to the path from x to y and it acts as a block that "mask" x to y node.

In Figure 5.7(c) the joint probability $\boldsymbol{p}(x, y, z)$ can be expressed as

$$\boldsymbol{p}(x, y, z) = \boldsymbol{p}(z|x, y)\boldsymbol{p}(x)\boldsymbol{p}(y), \tag{5.67}$$

and in this case the marginalization of z gives

$$\boldsymbol{p}(x, y) = \int_z \boldsymbol{p}(z|x, y)\boldsymbol{p}(x)\boldsymbol{p}(y) = \boldsymbol{p}(x)\boldsymbol{p}(y), \tag{5.68}$$

thus, x and y are independent if z is not observed, while considering the conditional probability

$$\boldsymbol{p}(x, y|z) = \frac{\boldsymbol{p}(x, y, z)}{\boldsymbol{p}(z)} = \frac{\boldsymbol{p}(z|x, y)\boldsymbol{p}(x)\boldsymbol{p}(y)}{\boldsymbol{p}(z)} \neq \boldsymbol{p}(y|z)\boldsymbol{p}(x|z), \tag{5.69}$$

it can be noticed that x and y are not conditionally independent given z. The node z is said to be "head-to-head" with respect to x and y and it represents a "block" in the path x-y when z is not observed, while conditioning on z remove the block, introducing dependence between x and y.

Considering again Figure 5.7(c), the

$$\boldsymbol{p}(x, y|w) = \frac{\int_z \boldsymbol{p}(x, y, z, w)}{\boldsymbol{p}(w)} = \frac{\int_z \boldsymbol{p}(z|x, y)\boldsymbol{p}(x)\boldsymbol{p}(y)\boldsymbol{p}(w|z)}{\boldsymbol{p}(w)}, \tag{5.70}$$

shows that the nodes x and y are not conditionally independent given w. This can be generalized to any *descendant* node of z, i.e., to any node that in the path starting from an head-to-head node is connected only by directed arrows. This implies that x and y are independent if z or any descendant node of z is not observed.

5.6.1.2 d-separation on DBN

The generalization of the previous properties of the DBN is the d-separation property. Lets consider a generic DBN with three non intersecting sets of nodes A, B and C, $A \cap B = A \cap C = B \cap C = \emptyset$ whose union may not cover all the nodes N ($A \cup B \cup C \subseteq N$). The sets of nodes A and B are conditionally independent given the nodes C if considering all the possible path from any node in A to any node in B is *blocked*. A path v is said to be blocked if along the path

- $\exists$ a head-to-tail or a tail-to-tail nodes in C
- $\forall$ head-to-head node and its descendant are not in C

If all paths are blocked A is said to be d-separated and the conditional independence of A and B given C is guaranteed.

5.6.1.3 Conditional Independent Sub-mapping on a SLAM DBN

Lets consider the DBN in Figure 5.8 representing an example of a SLAM problem where the solution is the estimation of $\boldsymbol{p}(\mathbf{\Gamma}_{0:3}, \mathbf{y}_{1:5}|\mathbf{u}_{1:3}, \mathbf{z}_{1:8})$, as inputs $\mathbf{u}$ and observations $\mathbf{z}$ are known. Let suppose to stop the estimation at $\mathbf{\Gamma}_2$, corresponding to the estimation of $\boldsymbol{p}(\mathbf{\Gamma}_{0:2}, \mathbf{y}_{1:3}|\mathbf{u}_{1:2}, \mathbf{z}_{1:4})$ and to start a new map. For clarity of explanation, lets consider the *global* submaps case and to extend to the *local* submaps case later. Recalling that $\mathbf{\Gamma}$ represent the robot pose with respect to a world reference system, it is possible to start a new map with the initial robot pose estimation correspondent to $\mathbf{\Gamma}_2$, i.e., the last estimated pose. The landmarks observed by the last robot pose may be observed in the next steps, thus they constitute a good choice for the initialization of the new map. Thus, the new map starts as $\boldsymbol{p}(\mathbf{\Gamma}_2, \mathbf{y}_3|\mathbf{u}_{1:2}, \mathbf{z}_{1:4})$, that is a marginal distribution of $\boldsymbol{p}(\mathbf{\Gamma}_{0:2}, \mathbf{y}_{1:3}|\mathbf{u}_{1:2}, \mathbf{z}_{1:4})$. Then, the new input $\mathbf{u}_3$ and the measurements $\mathbf{z}_{5:8}$ produce the evolution of the estimate up to $\boldsymbol{p}(\mathbf{\Gamma}_{2:3}, \mathbf{y}_{3:5}|\mathbf{u}_{1:3}, \mathbf{z}_{1:8})$.

Lets consider the sets of nodes $A = \{\mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \mathbf{y}_1, \mathbf{y}_2\}$, $B = \{\mathbf{\Gamma}_3, \mathbf{y}_4, \mathbf{y}_5\}$ and $C = \{\mathbf{\Gamma}_3, \mathbf{y}_3\}$, $\mathbf{z}_A = \{\mathbf{z}_1, \dots, \mathbf{z}_4, \mathbf{u}_1, \mathbf{u}_2\}$, $\mathbf{z}_B = \{\mathbf{z}_5, \dots, \mathbf{z}_8, \mathbf{u}_3\}$. The only paths that go from nodes in A to B are blocked, thus the DBN is d-separated given the nodes in C. This implies that nodes in $A \cup \mathbf{z}_A$ are conditionally independent of nodes in $B \cup \mathbf{z}_B$ given the nodes in C. This property can be expressed by

$$\boldsymbol{p}(A|B, C, \mathbf{z}_A, \mathbf{z}_B) = \boldsymbol{p}(A|C, \mathbf{z}_A) \tag{5.71}$$

$$\boldsymbol{p}(B|A, C, \mathbf{z}_A, \mathbf{z}_B) = \boldsymbol{p}(B|C, \mathbf{z}_B) \tag{5.72}$$

$$\tag{5.73}$$

This implies that, if the nodes in C are known, the two submaps do not carry any additional information each other.

5.6.1.4 Local Submaps

The previous example shows that a DBN representing a SLAM problem can be thought in terms of global submaps that are conditionally independent. This solves the first issue related to EKF SLAM, i.e. the computational intractability of the problem when the explored area grows. The usage of global submaps does not resolve the second issue, related

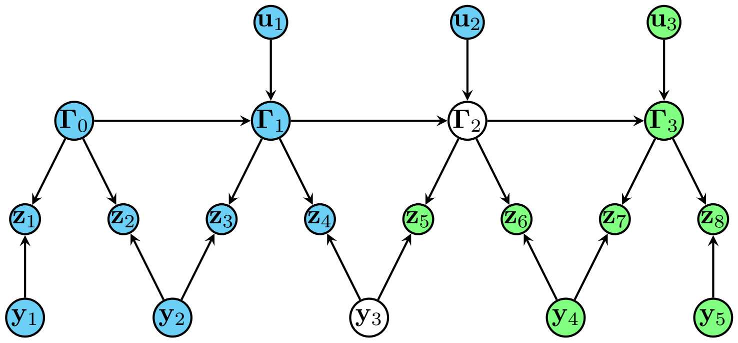


Figure 5.8: *A DBN of a SLAM problem where the conditional independence properties can be inferred considering the set of nodes* $(A \cup \mathbf{z}_A)$ *(in cyan),* $(B \cup \mathbf{z}_B)$ *(in green),* (C) *(in white), where* $A = \{\mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \mathbf{y}_1, \mathbf{y}_2\}$*,* $B = \{\mathbf{\Gamma}_3, \mathbf{y}_4, \mathbf{y}_5\}$ *and* $C = \{\mathbf{\Gamma}_3, \mathbf{y}_3\}$*,* $\mathbf{z}_A = \{\mathbf{z}_1, \dots, \mathbf{z}_4, \mathbf{u}_1, \mathbf{u}_2\}$*,* $\mathbf{z}_B = \{\mathbf{z}_5, \dots, \mathbf{z}_8, \mathbf{u}_3\}$*.*

to the amplification of approximations introduced by linearizations when the uncertainty grows. A better approach comes from the usage of local submaps, i.e., by starting each submap in a local origin. This implies that the (local) pose of the robot has zero uncertainty, thus landmarks uncertainty, that is influenced by the uncertainty in the robot pose, are smaller. This cuts down the effects of linearizion. The relative position of a map with respect to the previous one is represented by the last robot pose in the previous map and landmarks shared between the maps have to be transformed in the new reference frame thanks to the last robot position.

Lets consider the DBN of Figure 5.9. Globally, it expresses the probability

$$\boldsymbol{p}(\mathbf{\Gamma}^{(1)}_{0:2}, \mathbf{\Gamma}^{(2)}_{2:3}, \mathbf{y}^{(1)}_{1:3}, \mathbf{y}^{(2)}_{3:5} | \mathbf{u}_{1:3}, \mathbf{z}_{1:8}) \tag{5.74}$$

where superscript indicates the reference frame of each variable. The first submap is constructed up to $\mathbf{\Gamma}^{(1)}_2$ then the creation of the new map is started. Before this, $\mathbf{\Gamma}^{(1)}_2$ is used to transform the common landmarks between maps, generating in this case $\mathbf{y}^{(2)}_3$. This last element is the common part between the two maps and it ensure conditional independence of the groups $A \cup \mathbf{z}_A$ and $B \cup \mathbf{z}_B$ given the nodes in C, where $A = \{\mathbf{\Gamma}^{(1)}_0, \mathbf{\Gamma}^{(1)}_1, \mathbf{\Gamma}^{(1)}_2, \mathbf{y}^{(1)}_1, \mathbf{y}^{(1)}_2, \mathbf{y}^{(1)}_3\}$, $B = \{\mathbf{\Gamma}^{(2)}_2, \mathbf{\Gamma}^{(2)}_3, \mathbf{y}^{(2)}_4, \mathbf{y}^{(2)}_5\}$ and $C = \{\mathbf{y}^{(2)}_3\}$, $\mathbf{z}_A = \{\mathbf{z}_1, \dots, \mathbf{z}_4, \mathbf{u}_1, \mathbf{u}_2\}$, $\mathbf{z}_B = \{\mathbf{z}_5, \dots, \mathbf{z}_8, \mathbf{u}_3\}$. Thus, the first submap estimates

$$\boldsymbol{p}(\mathbf{\Gamma}^{(1)}_{0:2}, \mathbf{y}^{(1)}_{1:3}, \mathbf{y}^{(2)}_3 | \mathbf{u}_{1:2}, \mathbf{z}_{1:4}), \tag{5.75}$$

the second submap starts by marginalizing $\mathbf{y}^{(2)}_3$ with the robot pose in the origin without uncertainty

$$\boldsymbol{p}(\mathbf{\Gamma}^{(2)}_2, \mathbf{y}^{(2)}_3 | \mathbf{u}_{1:2}, \mathbf{z}_{1:4}), \tag{5.76}$$

and it evolves estimation up to

$$\boldsymbol{p}(\mathbf{\Gamma}^{(2)}_{2:3}, \mathbf{y}^{(2)}_{3:5} | \mathbf{u}_{1:3}, \mathbf{z}_{1:8}). \tag{5.77}$$

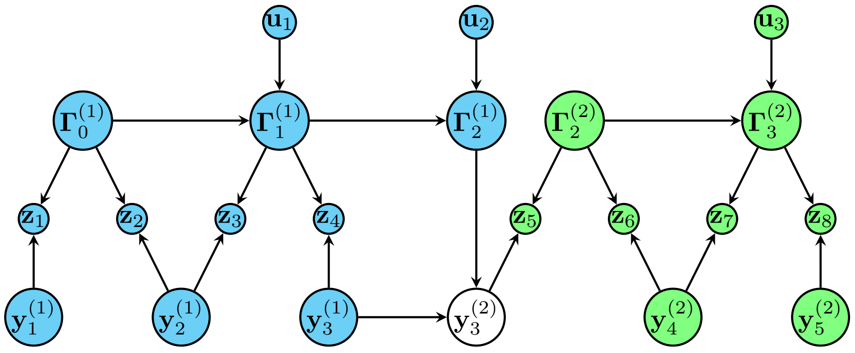


Figure 5.9: *A DBN of a SLAM problem where the conditional independence properties can be inferred considering the set of nodes $(A \cup \mathbf{z}_A)$, $(B \cup \mathbf{z}_B)$, (C), where $A = \{\mathbf{\Gamma}_0^{(1)}, \mathbf{\Gamma}_1^{(1)}, \mathbf{\Gamma}_2^{(1)}, \mathbf{y}_1^{(1)}, \mathbf{y}_2^{(1)}, \mathbf{y}_2^{(1)}\}$, $B = \{\mathbf{\Gamma}_2^{(2)}, \mathbf{\Gamma}_3^{(2)}, \mathbf{y}_4^{(2)}, \mathbf{y}_5^{(2)}\}$ and $C = \{\mathbf{y}_3^{(2)}\}$, $\mathbf{z}_A = \{\mathbf{z}_1, \dots, \mathbf{z}_4, \mathbf{u}_1, \mathbf{u}_2\}$, $\mathbf{z}_B = \{\mathbf{z}_5, \dots, \mathbf{z}_8, \mathbf{u}_3\}$. Notice that elements of the new submap are reffered to a different reference system, indicated in the superscript. The landmark $\mathbf{y}_3^{(1)}$, expressed in the reference system of the first submap, is transformed through $\mathbf{\Gamma}_2^{(1)}$ in $\mathbf{y}_3^{(2)}$, expressed in the reference system of the new submap. Notice that it is necessary to share only landmarks between maps, while the robot poses are not needed.*

5.6.1.5 Recovering the Whole Map

Lets rewrite the submaps probability in terms of the sets A, B, C, $\mathbf{z}_A$, $\mathbf{z}_B$; notice that this formulation allows to treat equally both the case of global and local submaps. The first map corresponds to

$$\boldsymbol{p}(A, C|\mathbf{z}_A), \tag{5.78}$$

while the second map corresponds to

$$\boldsymbol{p}(B, C|\mathbf{z}_A, \mathbf{z}_B). \tag{5.79}$$

The whole map estimation corresponds to the estimation of

$$\boldsymbol{p}(A, B, C|\mathbf{z}_A, \mathbf{z}_B). \tag{5.80}$$

where, when considering local submaps, it has to take in account that elements are expressed in different reference systems, thus some additional operation has to be performed if a the whole map has to be expressed in a unique reference system. The whole map can be partitioned as

$$\boldsymbol{p}(A, B, C|\mathbf{z}_A, \mathbf{z}_B) \quad = \quad \boldsymbol{p}(A|B, C, \mathbf{z}_A, \mathbf{z}_B)\boldsymbol{p}(B, C|\mathbf{z}_A, \mathbf{z}_B), \tag{5.81}$$

and, thanks to conditional independence (Equation 4.34) it is equal to

$$\boldsymbol{p}(A, B, C|\mathbf{z}_A, \mathbf{z}_B) \quad = \quad \boldsymbol{p}(A|C, \mathbf{z}_A)\boldsymbol{p}(B, C|\mathbf{z}_A, \mathbf{z}_B). \tag{5.82}$$

The second terms of the right hand side is exactly the estimation of the second submap, while the first terms can be obtained, by conditioning on C as

$$\boldsymbol{p}(A|C, \mathbf{z}_A) = \frac{\boldsymbol{p}(A, C|\mathbf{z}_A)}{\boldsymbol{p}(C|\mathbf{z}_A)}. \tag{5.83}$$

By combining Equations 5.82 and 5.83, the global map distribution results

$$\boldsymbol{p}(A, B, C|\mathbf{z}_A, \mathbf{z}_B) = \frac{\boldsymbol{p}(A, C|\mathbf{z}_A)}{\boldsymbol{p}(C|\mathbf{z}_A)}\boldsymbol{p}(B, C|\mathbf{z}_A, \mathbf{z}_B). \tag{5.84}$$

5.6.1.6 Loop Closure

Lets suppose the case in which landmarks regarding the map i are measured by a map $j > i$. This case is commonly referred as *loop closure*, i.e., the robot is taking measures from an already estimated area of the environment, and this needs a special attention in order to be treated within the conditional independent submapping techniques. To maintain the conditional independence property, the subset of landmarks L of map i that are involved in a loop closure have to become part of the shared element between all maps from i to j. This implies, in case of local submaps, to generate a transformed copy of each landmark in L.

Lets consider the case of Figure 5.10. The landmark $\mathbf{y}_1$ is observed in map 3 with measurement z_{14}. To maintain conditional independence $\mathbf{y}_1^{(2)}$ is generated from $\mathbf{y}_1^{(1)}$ and $\mathbf{\Gamma}_1^{(1)}$, and then shared between map 1 and 2. Then, $\mathbf{y}_1^{(2)}$ and $\mathbf{\Gamma}_1^{(2)}$ generate $\mathbf{y}_1^{(3)}$ that is shared between maps 2 and 3 and measured by z_{14}. Notice that if some landmark involved in the loop closure is already shared between the map i and some successive map k, they have to be copied and transformed only in maps $k + 1$ to j.

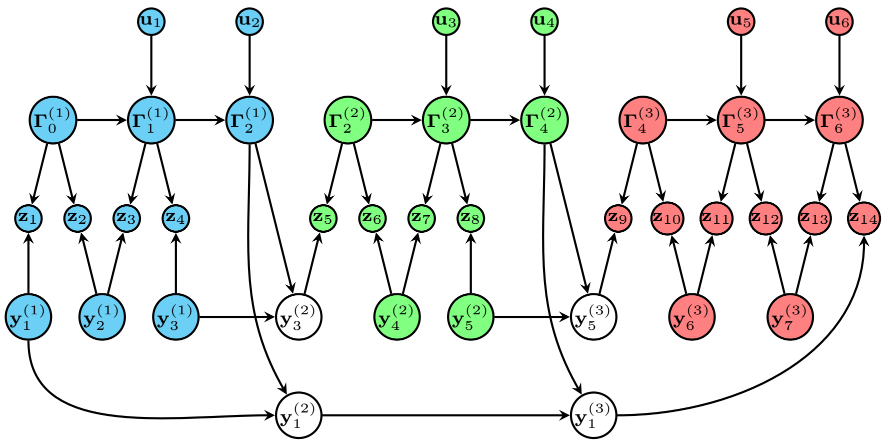
5.6.2 Map Transition Step

Lets now consider how to apply the theoretic concepts and formulas presented in the previous section to the EKF SLAM algorithm. In particular we have to define the three main steps that allow to implement the CI SLAM framework: *Map Transition*, *Back Propagation* and *Loop Closure*.

To describe the Map Transition Step, consider an EKF SLAM with vector state ${}^{\{i\}}\mathbf{x}$ summarized by the mean vector ${}^{\{i\}}\boldsymbol{\mu}$ and the covariance matrix ${}^{\{i\}}\mathbf{\Sigma}$, where the bracketed left superscript i indicates the current sub map number while the common subscript k that indicate time step is discarded in order to keep compact the notation. Consider that the state vector is composed by the robot pose ${}^{\{i\}}\mathbf{\Gamma}$ expressed with respect to the i-th map reference frame, optionally the vector of motion parameters ${}^{\{i\}}\mathbf{\Lambda}$ and by the set of landmarks $\{{}^{\{i\}}\mathbf{y}_j\}$ with $j \in [1, n]$:

$${}^{\{i\}}\mathbf{x} = \left[{}^{\{i\}}\mathbf{\Gamma}^T,\ {}^{\{i\}}\mathbf{\Lambda}^T,\ {}^{\{i\}}\mathbf{y}_1{}^T, \ldots, {}^{\{i\}}\mathbf{y}_n{}^T\right]^T, \tag{5.85}$$

Let assume that the current map i needs to be closed and a new map $i+1$ has to be started. Map closure criterion can be, for instance, the reaching of a threshold on the number of landmarks $n \geq n_{max}$ and consider a subset of landmarks $\{{}^{\{i\}}\mathbf{y}_S\} \subseteq \{{}^{\{i\}}\mathbf{y}_{1:n}\}$ that are



$\mathbf{u}_1$
$\mathbf{u}_2$
$\mathbf{u}_3$
$\mathbf{u}_4$
$\mathbf{u}_5$
$\mathbf{u}_6$
$\mathbf{\Gamma}_0^{(1)}$
$\mathbf{\Gamma}_1^{(1)}$
$\mathbf{\Gamma}_2^{(1)}$
$\mathbf{\Gamma}_2^{(2)}$
$\mathbf{\Gamma}_3^{(2)}$
$\mathbf{\Gamma}_4^{(2)}$
$\mathbf{\Gamma}_4^{(3)}$
$\mathbf{\Gamma}_5^{(3)}$
$\mathbf{\Gamma}_6^{(3)}$
$\mathbf{z}_1$ $\mathbf{z}_2$ $\mathbf{z}_3$ $\mathbf{z}_4$ $\mathbf{z}_5$ $\mathbf{z}_6$ $\mathbf{z}_7$ $\mathbf{z}_8$ $\mathbf{z}_9$ $\mathbf{z}_{10}$ $\mathbf{z}_{11}$ $\mathbf{z}_{12}$ $\mathbf{z}_{13}$ $\mathbf{z}_{14}$
$\mathbf{y}_1^{(1)}$ $\mathbf{y}_2^{(1)}$ $\mathbf{y}_3^{(1)}$ $\mathbf{y}_3^{(2)}$ $\mathbf{y}_4^{(2)}$ $\mathbf{y}_5^{(2)}$ $\mathbf{y}_5^{(3)}$ $\mathbf{y}_6^{(3)}$ $\mathbf{y}_7^{(3)}$
$\mathbf{y}_1^{(2)}$ $\mathbf{y}_1^{(3)}$

Figure 5.10: *A DBN of a SLAM problem where a loop closure is performed on landmark* $\mathbf{y}_1^{(1)}$*. The* $\mathbf{y}_1^{(2)}$ *is generated and is shared between map* 1 *and* 2*, then* $\mathbf{y}_1^{(2)}$ *generates* $\mathbf{y}_1^{(3)}$ *that is shared with map* 3 *and allows to performs measure* z_{14} *that ensure the loop closure.*

selected to be *shared* between map i and $i+1$, where $S = \{s_1, s_2, \dots, s_m\} \subseteq 1 : n$ is the set of the indexes of the selected landmarks. A standard criterion for the choice of landmarks to be shared is to select the landmarks that had been used in the last update step, since it is likely that they will be measured in the next steps.

5.6.2.1 LandmarkTransformation in the Current Map

The landmarks $\{{}^{\{i\}}\mathbf{y}_S\}$ need to be copied into the new map to initialize the system and to allow the continuation of the SLAM algorithm, but these landmarks are expressed in the reference frame of the map i, thus they need to be referenced to the $i+1$ map reference system before they can be copied. Let enhance the notation of landmarks ${}^{\{i\}}\mathbf{y}$ to ${}^{\{i\}}\mathbf{y}^{(i)}$, where the bracketed right superscript indicates the reference system in which the landmark is expressed. A transformed version of the landmark referred to the $i+1$ map reference system can be computed as

$$
{}^{\{i\}}\mathbf{y}^{(i+1)} \quad = \quad lt(\mathbf{T}_{(i)}^{(i+1)}, \; {}^{\{i\}}\mathbf{y}^{(i)}), \tag{5.86}
$$

where the function lt apply the transformation $\mathbf{T}_{(i)}^{(i+1)}$, i.e., the rototranslation that express the position of reference system i expressed in coordinate of $i+1$, to the landmarks expressed in reference system i, obtaining the landmarks in coordinate of reference system $i+1$. For instance, if landmarks were simple 3D points, the $lt(\cdots)$ function would be the standard point transformation equation (Equation 2.15). The transformation $\mathbf{T}_{(i)}^{(i+1)}$ has to be retrieved by the inversion of ${}^{\{i\}}\mathbf{\Gamma}$, as

$$
\mathbf{T}_{(i)}^{(i+1)} \quad \equiv \quad {}^{\{i\}}\mathbf{\Gamma}^{-1}, \tag{5.87}
$$

i.e., the last robot pose in the map i is the pose of the reference system $i+1$ expressed in the reference frame of map i.

The set of landmarks $\{{}^{\{i\}}\mathbf{y}_S^{(i+1)}\}$, obtained by the transformation of $\{{}^{\{i\}}\mathbf{y}_S^{(i)}\}$ with function $lt(\cdots)$, has to be added to the vector state of the EKF representing the map i. Thus, the mean values of the new landmarks can be computed by evaluating the function $lt(\cdots)$ in the current mean values of the state vector, while the new covariance matrix has to be computed with a procedure similar to the one used for the landmark addition presented in Section 5.4.4.

Let consider the landmark with index $s_j \in S$ has to be transformed and all the landmarks with indexes $s_1 : s_{j-1}$ were already transformed. The state vector, and consequently the mean vector, can be partitioned as it follows

$$\begin{aligned}{}^{\{i\}}\boldsymbol{\mu} &= \left[{}^{\{i\}}\bar{\boldsymbol{\Gamma}}^T,\ {}^{\{i\}}\bar{\boldsymbol{\Lambda}}^T,\ {}^{\{i\}}\bar{\mathbf{y}}_1^{(i)}{}^T,\dots,{}^{\{i\}}\bar{\mathbf{y}}_{s_j}^{(i)}{}^T,\dots,{}^{\{i\}}\bar{\mathbf{y}}_n^{(i)}{}^T,\ {}^{\{i\}}\bar{\mathbf{y}}_{s_1}^{(i+1)}{}^T,\dots,{}^{\{i\}}\bar{\mathbf{y}}_{s_{j-1}}^{(i+1)}{}^T\right]^T\\ &= \left[{}^{\{i\}}\bar{\boldsymbol{\Gamma}}^T,\ \bar{\boldsymbol{\mu}}_a^T,\ {}^{\{i\}}\bar{\mathbf{y}}_{s_j}^{(i)}{}^T,\bar{\boldsymbol{\mu}}_b^T\right]^T,\end{aligned} \tag{5.88}$$

where the landmark ${}^{\{i\}}\mathbf{y}_{s_j} \in \{{}^{\{i\}}\mathbf{y}_S\}$, i.e., it is the landmark to transform, the vector ${}^{\{i\}}\bar{\boldsymbol{\mu}}_a$ enclose the (optional) element ${}^{\{i\}}\bar{\boldsymbol{\Lambda}}^{(i)}$ and all the landmarks ${}^{\{i\}}\bar{\mathbf{y}}_{1:s_j-1}^T$ and the vector ${}^{\{i\}}\bar{\boldsymbol{\mu}}_b$ encloses all remaining part of the mean vector, i.e., the landmarks ${}^{\{i\}}\bar{\mathbf{y}}_{s_j+1:n}^T$ and the already transformed landmarks ${}^{\{i+1\}}\bar{\mathbf{y}}_{s_{1:j-1}}^T$. The partitioned covariance matrix is

$${}^{\{i\}}\boldsymbol{\Sigma} = \begin{bmatrix} {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}A} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}s_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}B}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}A}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{AA} & {}^{\{i\}}\boldsymbol{\Sigma}_{As_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{AB}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}s_j}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{As_j}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{s_js_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{s_jB}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}B}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{AB}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{s_jB}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{BB}\end{bmatrix}. \tag{5.89}$$

The addition of the transformed landmark ${}^{\{i+1\}}\mathbf{y}_{s_j}$ is done by appending

$${}^{\{i+1\}}\bar{\mathbf{y}}_{s_j} = lt({}^{\{i\}}\bar{\boldsymbol{\Gamma}}^{-1},{}^{\{i+1\}}\bar{\mathbf{y}}_{s_j}), \tag{5.90}$$

to the mean vector, resulting in

$${}^{\{i\}}\boldsymbol{\mu}^* = \left[{}^{\{i\}}\bar{\boldsymbol{\Gamma}}^T,\ \bar{\boldsymbol{\mu}}_a^T,\ {}^{\{i\}}\bar{\mathbf{y}}_{s_j}^T,\bar{\boldsymbol{\mu}}_b^T,\ {}^{\{i+1\}}\bar{\mathbf{y}}_{s_j}\right]^T, \tag{5.91}$$

and by enlarging the covariance matrix thanks to the Jacobians of $lt(\cdots)$ function

$${}^{\{i\}}\boldsymbol{\Sigma}^* = \begin{bmatrix}\mathbf{I} & 0 & 0 & 0\\ 0 & \mathbf{I} & 0 & 0\\ 0 & 0 & \mathbf{I} & 0\\ 0 & 0 & 0 & \mathbf{I}\\ \mathbf{J}_{\boldsymbol{\Gamma}} & 0 & \mathbf{J}_{\mathbf{y}} & 0\end{bmatrix} \begin{bmatrix} {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}A} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}s_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}B}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}A}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{AA} & {}^{\{i\}}\boldsymbol{\Sigma}_{As_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{AB}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}s_j}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{As_j}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{s_js_j} & {}^{\{i\}}\boldsymbol{\Sigma}_{s_jB}\\ {}^{\{i\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}B}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{AB}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{s_jB}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{BB}\end{bmatrix} \begin{bmatrix}\mathbf{I} & 0 & 0 & 0 & \mathbf{J}_{\boldsymbol{\Gamma}}^T\\ 0 & \mathbf{I} & 0 & 0 & 0\\ 0 & 0 & \mathbf{I} & 0 & \mathbf{J}_{\mathbf{y}}^T\\ 0 & 0 & 0 & \mathbf{I} & 0\end{bmatrix}, \tag{5.92}$$

where

$$\mathbf{J}_{\mathbf{\Gamma}} = \left.\frac{\partial lt(\mathbf{\Gamma}^{-1}, \mathbf{y})}{\partial \mathbf{\Gamma}}\right|_{\mathbf{\Gamma}={}^{\{i\}}\bar{\mathbf{\Gamma}}, \mathbf{y}={}^{\{i\}}\bar{\mathbf{y}}_{s_j}}, \tag{5.93}$$

$$\mathbf{J}_{\mathbf{y}} = \left.\frac{\partial lt(\mathbf{\Gamma}^{-1}, \mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{\Gamma}={}^{\{i\}}\bar{\mathbf{\Gamma}}, \mathbf{y}={}^{\{i\}}\bar{\mathbf{y}}_{s_j}}, \tag{5.94}$$

are the Jacobians evaluated at the current mean values of the estimation. As for the landmark addition procedure, the resulting covariance matrix ${}^{\{i\}}\mathbf{\Sigma}^*$ is an enlarged matrix that differs from ${}^{\{i\}}\mathbf{\Sigma}$ only in the added right band and in the added bottom band, while the top left block, containing the entire ${}^{\{i\}}\mathbf{\Sigma}$ is leaved unchanged. This allow to transform a single landmark in $O(n)$ time, where n is the total number of landmarks, thus the transformation of all the shared landmark $\{{}^{\{i\}}\mathbf{y}_S\}$ has a computational cost of $O(nm)$ where m, i.e., the number of landmarks to be shared, is obviously lower than n.

5.6.2.2 Parameter Transformation in the Current Map

The optional element $\mathbf{\Lambda}$ in the state vector can be present to specify some motion parameters, like the robot translational and rotational speed. These elements needs to be shared when a map transition is performed. If they are expressed with respect to the map reference system, they need to be transformed similarly to the landmarks. Naming $pt(\cdots)$ the transformation function, we have to follow the same procedure explained above to transform the parameters thanks to

$${}^{\{i\}}\bar{\mathbf{\Lambda}}^{(i+1)} = pt({}^{\{i\}}\bar{\mathbf{\Gamma}}^{-1}, {}^{\{i\}}\bar{\mathbf{\Lambda}}^{(i)}), \tag{5.95}$$

to enlarge the mean vector with the ${}^{\{i\}}\bar{\mathbf{\Lambda}}^{(i+1)}$ elements and to enlarge the covariance matrix through Jacobians of the transformation function.

5.6.2.3 New Map Initialization

The new map (i.e., the $i+1$ map), alike in a new EKF SLAM system, starts in the origin $\mathbf{\Gamma}_0$ with zero covariance, since it is a *local map*. If it is present, the motion parameters variable ${}^{\{i+1\}}\mathbf{\Lambda}$ has to be copied from the i-th map. In case motion parameters are expressed in robot reference frame, the variable ${}^{\{i+1\}}\mathbf{\Lambda}$ is copied from ${}^{\{i\}}\mathbf{\Lambda}$, while if it is expressed in the map reference frame, it has to be copied from the transformed element ${}^{\{i\}}\mathbf{\Lambda}^{(i+1)}$. Moreover, the motion parameters variable has to be copied twice: the first copy (${}^{\{i+1\}}\mathbf{\Lambda}$) will be used in the motion model, i.e., it evolutes in time, while the second copy (${}^{\{i\}}\mathbf{\Lambda}_s$) will not evolve in the prediction step and it will be only affected by the update step. In fact, it represents the value of the motion parameters at the time in which the new map was started and the previous one was closed. Finally, all the landmarks that are selected to be shared have to be copied as ${}^{\{i+1\}}\mathbf{y}_j{}^{(i+1)} = {}^{\{i\}}\mathbf{y}_{s_j}{}^{(i+1)}$ where $j = 1 : m$ being m the number of elements of the set S. Thus, the new state vector results

$$^{\{i+1\}}\mathbf{x} = \begin{bmatrix} ^{\{i+1\}}\mathbf{\Gamma} \\ ^{\{i+1\}}\mathbf{\Lambda} \\ ^{\{i+1\}}\mathbf{\Lambda}_s \\ ^{\{i+1\}}\mathbf{y}_1 \\ \vdots \\ ^{\{i+1\}}\mathbf{y}_m \end{bmatrix} = \begin{bmatrix} ^{\{i+1\}}\mathbf{\Gamma}_0 \\ ^{\{i\}}\mathbf{\Lambda}^{(i+1)} \text{ or } ^{\{i\}}\mathbf{\Lambda} \\ ^{\{i\}}\mathbf{\Lambda}^{(i+1)} \text{ or } ^{\{i\}}\mathbf{\Lambda} \\ ^{\{i\}}\mathbf{y}_{s_1}{}^{(i+1)} \\ \vdots \\ ^{\{i\}}\mathbf{y}_{s_m}{}^{(i+1)} \end{bmatrix}. \tag{5.96}$$

To copy variables from the EKF that represents the map i to the $i+1$, both mean values and covariances of the transformed landmarks (and eventually the motion parameters element) have to be copied. This is done by selecting the corresponding rows in the mean values vector and the corresponding rows and columns in the covariance matrix of the map i (i.e., by marginalize out the variables to be copied) and to copy them in the new EKF mean vector and covariance matrix, taking into account that the top band and the left band of the covariance matrix is zero, since the initial pose of the new map has zero covariance and it is initially uncorrelated with the rest of the variables.

5.6.2.4 Shared Elements Between Maps

Correspondences between the variables in the maps are needed by the *back propagation step*, thus they have to be stored. Lets name $SP = \{\langle ^{\{a\}}\mathbf{v}, ^{\{b\}}\mathbf{w}\rangle\}$ the entire set of pair of variables that correspond between maps, where a and b are respectively the source and the destination map index, and $\mathbf{v}$ and $\mathbf{w}$ the two corresponding variables in the EKF of the two maps. When a new map $i+1$ is created, this set has to be enlarged with the pairs of variables shared between the two maps. It is important to notice that the shared variables are not necessarily all the variables that have been copied. In particular, if the parameters variable $^{\{i\}}\mathbf{\Lambda}$ is present in the original map, only its second copy $^{\{i+1\}}\mathbf{\Lambda}_s$ represents a shared variable. This comes from the fact that the first copy $^{\{i+1\}}\mathbf{\Lambda}$ does not belong to the elements that ensure the conditional independence of the map, but it is only used to initialize the state of the system properly, i.e., to give to the motion model the right informations on the robot motion when the map has been started. Thus, the correspondences set is enlarged by

$$SP \leftarrow SP \cup \left\{ \begin{array}{rcl} \left\langle ^{\{i\}}\mathbf{\Lambda}^{(i+1)} \text{ or } ^{\{i\}}\mathbf{\Lambda} \right. & , & \left. ^{\{i+1\}}\mathbf{\Lambda}_s \right\rangle \\ \langle ^{\{i\}}\mathbf{y}_{s_1}{}^{(i+1)} & , & ^{\{i+1\}}\mathbf{y}_1 \rangle \\ & \vdots & \\ \langle ^{\{i\}}\mathbf{y}_{s_m}{}^{(i+1)} & , & ^{\{i+1\}}\mathbf{y}_m \rangle \end{array} \right\}. \tag{5.97}$$

Once the map creation process is completed, the EKF slam algorithm proceeds with the standard steps, i.e., the motion prediction, the measurement step, the update step and the addition of the new landmarks. Differently from starting an EKF from scratch, the map transition starts a SLAM sub problem with an initialized structure.

5.6.3 Back-Propagation Step

The back-propagation step aims at the update of the closed maps thanks to the estimate obtained with the new observations gathered since the previous back-propagation step. Thus, starting from the last map i, it is possible to iterate back propagation between pairs of maps, i.e., to back propagate information from map i to map $i-1$, from map $i-1$ to map $i-2$ and so on until the first map. Let consider two maps i and $i-1$ and to partition their state vectors as

$$\begin{aligned} {}^{\{i-1\}}\mathbf{x} &= \left[{}^{\{i-1\}}\mathbf{x}_A,\ {}^{\{i-1\}}\mathbf{x}_C\right], & (5.98)\\ {}^{\{i\}}\mathbf{x} &= \left[{}^{\{i\}}\mathbf{x}_B,\ {}^{\{i\}}\mathbf{x}_C\right], & (5.99) \end{aligned}$$

where ${}^{\{i-1\}}\mathbf{x}_C$ and ${}^{\{i\}}\mathbf{x}_C$ are corresponding variables in the two maps, i.e., they have been shared when map i has been created starting from map $i-1$, ${}^{\{i-1\}}\mathbf{x}_A$ are the variables in the map $i-1$ that are not shared with the map i and ${}^{\{i\}}\mathbf{x}_B$ are the variable of the map i that are not shared with the map $i-1$. Mean vectors and covariance matrices results

$$\begin{aligned} {}^{\{i-1\}}\boldsymbol{\mu} &= \left[{}^{\{i-1\}}\boldsymbol{\mu}_A{}^T,\ {}^{\{i-1\}}\boldsymbol{\mu}_C{}^T\right]^T, & (5.100)\\ {}^{\{i-1\}}\boldsymbol{\Sigma} &= \begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\Sigma}_{AA} & {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC} \\ {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}{}^T & {}^{\{i-1\}}\boldsymbol{\Sigma}_{CC} \end{bmatrix}, & (5.101)\\ {}^{\{i\}}\boldsymbol{\mu} &= \left[{}^{\{i\}}\boldsymbol{\mu}_B{}^T,\ {}^{\{i\}}\boldsymbol{\mu}_C{}^T\right]^T, & (5.102)\\ {}^{\{i\}}\boldsymbol{\Sigma} &= \begin{bmatrix} {}^{\{i\}}\boldsymbol{\Sigma}_{BB} & {}^{\{i\}}\boldsymbol{\Sigma}_{BC} \\ {}^{\{i\}}\boldsymbol{\Sigma}_{BC}{}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{CC} \end{bmatrix}. & (5.103) \end{aligned}$$

The ${}^{\{i-1\}}\boldsymbol{\mu}_C$ and ${}^{\{i\}}\boldsymbol{\mu}_C$ variables set should have the same values in terms of mean vector and covariance matrices, but ${}^{\{i\}}\mathbf{x}_C$ has been estimated using more measures than ${}^{\{i-1\}}\mathbf{x}_C$. Thanks to conditional independence it is possible to update the map $i-1$ estimate as

$$\begin{aligned} {}^{\{i-1\}}\boldsymbol{\mu}^* &= \begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\mu}^*_A \\ {}^{\{i\}}\boldsymbol{\mu}_C \end{bmatrix}, & (5.104)\\ {}^{\{i-1\}}\boldsymbol{\Sigma}^* &= \begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AA} & {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} \\ {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC}{}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{CC} \end{bmatrix}, & (5.105) \end{aligned}$$

where

$$\begin{aligned} \mathbf{K} &= {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1} & (5.106)\\ {}^{\{i-1\}}\boldsymbol{\mu}^*_A &= {}^{\{i-1\}}\boldsymbol{\mu}_A + \mathbf{K}\left({}^{\{i\}}\boldsymbol{\mu}_C - {}^{\{i-1\}}\boldsymbol{\mu}_C\right) & (5.107)\\ {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} &= \mathbf{K}\,{}^{\{i\}}\boldsymbol{\Sigma}_{CC} & (5.108)\\ {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AA} &= {}^{\{i-1\}}\boldsymbol{\Sigma}_{AA} + \mathbf{K}\left({}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} - {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\right)^T & (5.109) \end{aligned}$$

It can be noticed that the shared element mean values and covariances are simply copied from the new map to the previous, while the non shared elements are updated thanks to a gain matrix $\mathbf{K}$.

The complexity of the back propagation between two maps is $O(1)$, since all the matrix operations are performed on a bounded number of landmarks. Consequently, the complete back propagation procedure has linear complexity in terms of the number of maps in the problem. It can be noticed that the back propagation step can be performed in parallel to the SLAM exploration process on a separate computing unit.

Back propagation needs a special care in the problem formulation to avoid the singularities in the covariance matrices, since in Equation 5.106 the matrix ${}^{\{i-1\}}\mathbf{\Sigma}_{CC}$, i.e., the covariance matrix of the shared elements in the previous map, has to be inverted. By definition, a covariance matrix is definite positive, but it results semidefinite positive if some linear relation exist between variables. To apply the Conditional Independent SLAM framework we have to ensure that the specific model for the robot pose and map representation in the EKF SLAM algorithm do not introduce singularities in the covariance matrix or, at least, in the elements that have to be shared between sub-maps.

5.6.3.1 Mathematical Derivation

The equations of the back propagation derive from the conditional independence properties. The joint probability of the whole map (Equation 5.80) $\boldsymbol{p}(A, B, C|\mathbf{z}_A, \mathbf{z}_B)$ is still a Gaussian variable, where mean vector and covariance matrix are

$$\boldsymbol{\mu}_{ABC} = \left[{}^{\{i-1\}}\boldsymbol{\mu}_A^{*\,T},\ \ {}^{\{i\}}\boldsymbol{\mu}_B{}^T,\ \ {}^{\{i\}}\boldsymbol{\mu}_C{}^T\right]^T, \tag{5.110}$$

$$\mathbf{\Sigma}_{ABC} = \begin{bmatrix} {}^{\{i-1\}}\mathbf{\Sigma}^*_{AA} & {}^{\{i-1\}}\mathbf{\Sigma}^*_{AB} & {}^{\{i-1\}}\mathbf{\Sigma}^*_{AC} \\ {}^{\{i-1\}}\mathbf{\Sigma}^{*\,T}_{AB} & {}^{\{i\}}\mathbf{\Sigma}_{BB} & {}^{\{i\}}\mathbf{\Sigma}_{BC} \\ {}^{\{i-1\}}\mathbf{\Sigma}^{*\,T}_{AC} & {}^{\{i\}}\mathbf{\Sigma}_{BC}{}^T & {}^{\{i\}}\mathbf{\Sigma}_{CC} \end{bmatrix}. \tag{5.111}$$

The last two block of the mean vector and the 2×2 bottom right block of the covariance matrix are exactly the estimation of the i-th map, i.e., the marginal distribution $\boldsymbol{p}(B, C|\mathbf{z}_A, \mathbf{z}_B)$, while the elements with the star in the superscript indicates the mean values and the covariances of the variables in the set A that belongs to the map $i-1$ but have been estimated with the measurement of $\mathbf{z}_B$ too. These elements are not directly known and the mathematical derivation of their values is in the following.

By marginalizing out the B variable, i.e., removing the second block of the mean vector and the second row and column of the covariance matrix, the parameters of $\boldsymbol{p}(A, C|\mathbf{z}_A, \mathbf{z}_B)$ are obtained. This can be considered as an updated version of the first map, since it integrates the measurements $\mathbf{z}_B$, while the original estimation of the map has been stopped early, using only the $\mathbf{z}_A$ measurements. By conditioning on the C variable it is possible to express $\boldsymbol{p}(A|C, \mathbf{z}_A, \mathbf{z}_B)$; the mean vector and covariance matrix of this distribution can be calculated with Equation 4.45, resulting in

$$\boldsymbol{\mu}_{A|C,\mathbf{z}_A,\mathbf{z}_B} = {}^{\{i-1\}}\boldsymbol{\mu}^\star_A + {}^{\{i-1\}}\mathbf{\Sigma}^*_{AC}\,{}^{\{i\}}\mathbf{\Sigma}_{CC}{}^{-1}\left(\mathbf{x}_C - {}^{\{i\}}\boldsymbol{\mu}_C\right), \tag{5.112}$$

$$\mathbf{\Sigma}_{A|C,\mathbf{z}_A,\mathbf{z}_B} = {}^{\{i-1\}}\mathbf{\Sigma}^*_{AA} - {}^{\{i-1\}}\mathbf{\Sigma}^*_{AC}\,{}^{\{i\}}\mathbf{\Sigma}_{CC}{}^{-1}\,{}^{\{i-1\}}\mathbf{\Sigma}^{*\,T}_{AC}. \tag{5.113}$$

The estimation of the first map, given by the mean vector and covariance matrix respectively of Equations 5.100 and 5.101, express $\boldsymbol{p}(A, C|\mathbf{z}_A)$. Alike the previous case, conditioning on C gives $\boldsymbol{p}(A|C,\mathbf{z}_A)$ with mean vector and covariance matrices

$$\begin{aligned}
\boldsymbol{\mu}_{A|C,\mathbf{z}_A} &= {}^{\{i-1\}}\boldsymbol{\mu}_A + {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\left(\mathbf{x}_C - {}^{\{i-1\}}\boldsymbol{\mu}_C\right), && (5.114)\\
\boldsymbol{\Sigma}_{A|C,\mathbf{z}_A} &= {}^{\{i-1\}}\boldsymbol{\Sigma}_{AA} - {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\boldsymbol{\Sigma}_{CC}{}^{-1}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}{}^{T}. && (5.115)
\end{aligned}$$

The conditional independence ensures that $\boldsymbol{p}(A|C,\mathbf{z}_A,\mathbf{z}_B) = \boldsymbol{p}(A|C,\mathbf{z}_A)$, thus, Equations 5.114 and 5.112 can be equated. By expanding the terms of the equivalence and after rewriting it with a more clear notation we obtain

$$\begin{aligned}
&\underbrace{{}^{\{i-1\}}\boldsymbol{\mu}^*_A}_{\mathbf{a}} + \underbrace{{}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC}}_{\mathbf{A}}\underbrace{{}^{\{i\}}\boldsymbol{\Sigma}_{CC}{}^{-1}}_{\mathbf{B}}\underbrace{\mathbf{x}_C}_{\mathbf{x}} - {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC}\,{}^{\{i\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\,\underbrace{{}^{\{i\}}\boldsymbol{\mu}_C}_{\mathbf{b}}\\
&= \underbrace{{}^{\{i-1\}}\boldsymbol{\mu}_A}_{\mathbf{c}} + \underbrace{{}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}}_{\mathbf{C}}\underbrace{{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}}_{\mathbf{D}}\mathbf{x}_C - {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\,\underbrace{{}^{\{i-1\}}\boldsymbol{\mu}_C}_{\mathbf{d}},
\end{aligned} \qquad (5.116)$$

$$\mathbf{a} - \mathbf{A}\mathbf{B}^{-1}\mathbf{b} + \mathbf{A}\mathbf{B}^{-1}\mathbf{x} = \mathbf{c} - \mathbf{C}\mathbf{D}^{-1}\mathbf{d} + \mathbf{C}\mathbf{D}^{-1}\mathbf{x}. \qquad (5.117)$$

This equation has to be valid for all $\mathbf{x}$ and the unknown are the vector $\mathbf{a}$ and the matrix $\mathbf{A}$ that correspond to the mean vector and to the covariance matrix of the shared elements between the map estimated with all the measurements. The identity can be analyzed separating the elements containing the $\mathbf{x}$ and the elements containing only constants, obtaining

$$\begin{aligned}
\mathbf{A}\mathbf{B}^{-1} &= \mathbf{C}\mathbf{D}^{-1}, && (5.118)\\
\mathbf{A} &= \mathbf{C}\mathbf{D}^{-1}\mathbf{B}, && (5.119)\\
{}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} &= {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\,{}^{\{i\}}\boldsymbol{\Sigma}_{CC}. && (5.120)
\end{aligned}$$

Putting Equation 5.118 in Equation 5.117 allows to retrieve $\mathbf{a} \equiv {}^{\{i-1\}}\boldsymbol{\mu}^*_A$ as

$$\begin{aligned}
\mathbf{a} &= \mathbf{c} + \mathbf{C}\mathbf{D}^{-1}(\mathbf{b} - \mathbf{d}), && (5.121)\\
{}^{\{i-1\}}\boldsymbol{\mu}^*_A &= {}^{\{i-1\}}\boldsymbol{\mu}_A + {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\left({}^{\{i\}}\boldsymbol{\mu}_C - {}^{\{i-1\}}\boldsymbol{\mu}_C\right). && (5.122)
\end{aligned}$$

From the Equations 5.115 and 5.113 it is possible to recover the last unknown term ${}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AA}$ as

$${}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AA} = {}^{\{i-1\}}\boldsymbol{\Sigma}_{AA} + {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\,{}^{\{i-1\}}\boldsymbol{\Sigma}_{CC}{}^{-1}\left({}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} - {}^{\{i-1\}}\boldsymbol{\Sigma}_{AC}\right)^T. \qquad (5.123)$$

It is possible to notice that Equations 5.120, 5.122 and 5.123 constitute the terms of the mean vector (Equation 5.104) and covariance (Equation 5.105) matrix after the back propagation step.

There is still a missing term in the matrix $\boldsymbol{\Sigma}_{ABC}$ of Equation 5.111, that is the covariance ${}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AB}$ of the variables A with the variables B. It can be noticed that this term is not required for the estimation with the submapping mechanism; the computation of this terms can be useful if two submaps have to be fused in a unique one. This term follows by conditioning $\boldsymbol{p}(A,B,C|\mathbf{z}_A,\mathbf{z}_B)$ on C, obtaining $\boldsymbol{p}(A,B|\mathbf{z}_A,\mathbf{z}_B,C)$ with the covariance matrix given by

$$\begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AA} & {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AB} \\ {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AB}{}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{BB} \end{bmatrix} - \begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} \\ {}^{\{i\}}\boldsymbol{\Sigma}_{BC} \end{bmatrix} {}^{\{i\}}\boldsymbol{\Sigma}_{CC}{}^{-1} \begin{bmatrix} {}^{\{i-1\}}\boldsymbol{\Sigma}^*_{AC} \\ {}^{\{i\}}\boldsymbol{\Sigma}_{BC} \end{bmatrix}^T. \qquad (5.124)$$

Due to the conditional independence of A and B given C, the non diagonal blocks have to be zero. This implies that

$$ {}^{\{i-1\}}\mathbf{\Sigma}^{*}_{AB}{}^{T} = {}^{\{i-1\}}\mathbf{\Sigma}^{*}_{AC}\,{}^{\{i\}}\mathbf{\Sigma}_{CC}{}^{-1}\,{}^{\{i\}}\mathbf{\Sigma}_{BC}, \tag{5.125} $$

that, thanks to the relation expressed in Equation 5.120 is equivalent to

$$ \begin{aligned} {}^{\{i-1\}}\mathbf{\Sigma}^{*}_{AB}{}^{T} &= {}^{\{i-1\}}\mathbf{\Sigma}_{AC}\,{}^{\{i-1\}}\mathbf{\Sigma}_{CC}{}^{-1}\,{}^{\{i\}}\mathbf{\Sigma}_{BC} && (5.126)\\ &= \mathbf{K}\,{}^{\{i\}}\mathbf{\Sigma}_{BC}. && (5.127) \end{aligned} $$

5.6.4 Loop closure

When a loop is detected, i.e., when some incoming observations measure landmarks that are already added to some map, the conditional independence framework needs to invoke a special procedure. If the landmarks subject to the loop closure are in the current working map, no special operation are required, since the loop closure is automatically handled by the standard EKF slam measurement and update mechanism. Differently, when the perceived landmarks are not in the current working map, but they are in a previous one, they have to be copied and passed by all the maps that links the original map with the current one. Once the landmarks are properly added to the last map a standard update step can be performed and, after a complete back propagation, all the maps benefit of the identified loop.

The procedure that allows to measure the landmarks involved in a loop closure between maps is detailed hereafter. Let consider the case in which a group of landmarks $\{\mathbf{y}_L\}$ in the map i are seen from map $j > i$. First of all it is necessary to perform a back propagation at least between map j and i, while the back propagation of maps previous to i can be postponed. Secondly it is necessary to identify for each landmark $\mathbf{y}_l$ with $l \in L$ the last map $m_l \geq i$ that contains already the landmark. Each landmark has to be added through maps $m_l + 1$ to j. This operation can be easily achieved by searching in the landmark correspondences pairs stored in SP.

Lets illustrate the procedure to add a set of landmarks $\{\mathbf{y}_L\}$ that are in the map i to the map $i+1$. The entire loop closure procedure is easily generalizable by iterating the addition to the successive maps, considering that the set $\{\mathbf{y}_L\}$ can grow if some landmark needs to be copied only on some successive maps. Each landmark ${}^{\{i\}}\mathbf{y}_l^{(i)}$ with $l \in L$ has to be transformed with the function $lt(\cdots)$, generating ${}^{\{i\}}\mathbf{y}_l^{(i+1)}$ and added to the state vector of the EKF with the same procedure of Section 5.6.2.1. Once all landmarks of $\{\mathbf{y}_L\}$ have been transformed, they have to be copied to the map $i+1$. Let consider the EKF mean vector and covariance matrix of the source map i and the destination map $i+1$, where the shared variable are added, partitioned as

$$
\begin{aligned}
{}^{\{i\}}\boldsymbol{\mu} &= \begin{bmatrix} {}^{\{i\}}\boldsymbol{\mu}_A^T & {}^{\{i\}}\boldsymbol{\mu}_C^T & {}^{\{i\}}\boldsymbol{\mu}_L^T \end{bmatrix}^T && (5.128)\\
{}^{\{i\}}\boldsymbol{\Sigma} &= \begin{bmatrix} {}^{\{i\}}\boldsymbol{\Sigma}_{AA} & {}^{\{i\}}\boldsymbol{\Sigma}_{AC} & {}^{\{i\}}\boldsymbol{\Sigma}_{AL} \\ {}^{\{i\}}\boldsymbol{\Sigma}_{AC}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{CC} & {}^{\{i\}}\boldsymbol{\Sigma}_{CL} \\ {}^{\{i\}}\boldsymbol{\Sigma}_{AL}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{CL}^T & {}^{\{i\}}\boldsymbol{\Sigma}_{LL} \end{bmatrix} && (5.129)\\
{}^{\{i+1\}}\boldsymbol{\mu} &= \begin{bmatrix} {}^{\{i+1\}}\boldsymbol{\mu}_B^T & {}^{\{i+1\}}\boldsymbol{\mu}_C^T & {}^{\{i+1\}}\boldsymbol{\mu}_L^T \end{bmatrix}^T && (5.130)\\
{}^{\{i+1\}}\boldsymbol{\Sigma} &= \begin{bmatrix} {}^{\{i+1\}}\boldsymbol{\Sigma}_{BB} & {}^{\{i+1\}}\boldsymbol{\Sigma}_{BC} & {}^{\{i+1\}}\boldsymbol{\Sigma}_{BL} \\ {}^{\{i+1\}}\boldsymbol{\Sigma}_{BC}^T & {}^{\{i+1\}}\boldsymbol{\Sigma}_{CC} & {}^{\{i+1\}}\boldsymbol{\Sigma}_{CL} \\ {}^{\{i+1\}}\boldsymbol{\Sigma}_{BL}^T & {}^{\{i+1\}}\boldsymbol{\Sigma}_{CL}^T & {}^{\{i+1\}}\boldsymbol{\Sigma}_{LL} \end{bmatrix} && (5.131)
\end{aligned}
$$

where the subscript A indicates the variables of map i that are not shared with $i + 1$, C the variables shared between the two maps and L the variables that are forming the loop closure. In particular, ${}^{\{i\}}\boldsymbol{\mu}_L^T$ and the bottom band and right band of ${}^{\{i\}}\boldsymbol{\Sigma}$ are the transformed landmarks in the map i that have to be copied to map $i + 1$. Since a back propagation step has been performed, the next equivalences are true:

$$
\begin{aligned}
{}^{\{i+1\}}\boldsymbol{\mu}_C &= {}^{\{i\}}\boldsymbol{\mu}_C, && (5.132)\\
{}^{\{i+1\}}\boldsymbol{\Sigma}_{CC} &= {}^{\{i\}}\boldsymbol{\Sigma}_{CC}, && (5.133)\\
{}^{\{i+1\}}\boldsymbol{\mu}_L &= {}^{\{i\}}\boldsymbol{\mu}_L, && (5.134)\\
{}^{\{i+1\}}\boldsymbol{\Sigma}_{LL} &= {}^{\{i\}}\boldsymbol{\Sigma}_{LL}, && (5.135)\\
{}^{\{i+1\}}\boldsymbol{\Sigma}_{CL} &= {}^{\{i\}}\boldsymbol{\Sigma}_{CL}. && (5.136)
\end{aligned}
$$

This means that the new element in the map $i + 1$ (the ones with the L subscript) are simply copied, apart from the ${}^{\{i+1\}}\boldsymbol{\Sigma}_{BL}$. The latter has to be computed leveraging on the covariance independence properties by conditioning $\boldsymbol{p}(B, L|C, \mathbf{z}_A, \mathbf{z}_B)$ and by imposing that the covariance between B and L is zero given C and the measurements:

$$
{}^{\{i+1\}}\boldsymbol{\Sigma}_{BL}^T = {}^{\{i+1\}}\boldsymbol{\Sigma}_{CL}^T \, {}^{\{i+1\}}\boldsymbol{\Sigma}_{CC}^{-1} \, {}^{\{i+1\}}\boldsymbol{\Sigma}_{BC}^T \qquad (5.137)
$$

Finally, the set of correspondences SP has to be updated by adding the new landmarks as shared variables between the two considered maps.

5.6.5 Summary With An Example

To conclude the presentation of the CI SLAM framework, we reproduce the illustrative example treated in Section 5.6.1.3 in a more practical way, considering the generation of the submaps, the back propagation step and a loop closure. In particular, we follow the evolution of the DBN depicted in Figure 5.9, but, in this case, we discard the input $\mathbf{u}_k$ and we assume to describe the robot dynamic state with the $\boldsymbol{\Lambda}$ element in the robot reference system.

When the system starts we have to create the first map, that is empty and initialized with the robot placed in the origin and with the initial values for the robot motion description. To be general, we consider a complete covariance matrix, although we know that some elements are zero at initialization time. The complete state of the first map is, at time 0,

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_0 &= \left[{}^{\{1\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{1\}}\bar{\boldsymbol{\Lambda}}^{T} \right]^T, &(5.138)\\
{}^{\{1\}}\boldsymbol{\Sigma}_0 &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Gamma}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Lambda}} \\ {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda\Lambda}} \end{bmatrix}. &(5.139)
\end{aligned}
$$

Since no landmark exists in the map, the sensor observation triggers the addition of new landmarks to the map. Let consider the new landmarks are ${}^{\{1\}}y_1^{(1)}$ and ${}^{\{1\}}y_2^{(1)}$, expressed w.r.t. the current map reference frame. The EKF state is enlarged, according to the addition step of the EKF SLAM algorithm as

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_0 &= \left[{}^{\{1\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{1\}}\bar{\boldsymbol{\Lambda}}^{T},\ {}^{\{1\}}\bar{\mathbf{y}}_1^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_2^{(1)T} \right]^T, &(5.140)\\
{}^{\{1\}}\boldsymbol{\Sigma}_0 &= \begin{bmatrix}
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Gamma}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_2} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Lambda}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Lambda}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\mathbf{y}_1\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_2}
\end{bmatrix}. &(5.141)
\end{aligned}
$$

Subsequently, motion prediction is performed and the landmark $\mathbf{y}_2$ is observed, generating an update step. Moreover, the landmark $\mathbf{y}_3$ is perceived and added to the map. The result of all these steps is the evolution of the EKF state as

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_1 &= \left[{}^{\{1\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{1\}}\bar{\boldsymbol{\Lambda}}^{T},\ {}^{\{1\}}\bar{\mathbf{y}}_1^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_2^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_3^{(1)T} \right]^T, &(5.142)\\
{}^{\{1\}}\boldsymbol{\Sigma}_1 &= \begin{bmatrix}
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Gamma}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Lambda}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Lambda}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\mathbf{y}_1\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3} \\
{}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Gamma}\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\boldsymbol{\Lambda}\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\mathbf{y}_1\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}^T_{\mathbf{y}_2\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3}
\end{bmatrix}. &(5.143)
\end{aligned}
$$

A new motion prediction is performed, thus the mean vector and the covariance matrix are coherently modified and they results in ${}^{\{1\}}\boldsymbol{\mu}_2$ and ${}^{\{1\}}\boldsymbol{\Sigma}_2$. At this time, before performing the updates, we want to close the first map and to start the second one sharing the landmark $\mathbf{y}_3$. We have to transform the ${}^{\{1\}}\mathbf{y}_3{}^{(1)}$ to ${}^{\{1\}}\mathbf{y}_3{}^{(2)}$ element with Equation 5.86

and add it to the EKF state of the first map, that results

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_2 &= \left[{}^{\{1\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{1\}}\bar{\boldsymbol{\Lambda}}^{T},\ {}^{\{1\}}\bar{\mathbf{y}}_1^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_2^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_3^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_3^{(2)T} \right]^T, \\
{}^{\{1\}}\boldsymbol{\Sigma}_2 &= \begin{bmatrix}
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3^{(2)}} \\
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3^{(2)}} \\
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_1}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3^{(2)}} \\
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_2}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3^{(2)}} \\
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3^{(2)}} \\
{}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3^{(2)}\mathbf{y}_3^{(2)}}
\end{bmatrix}.
\end{aligned}
\tag{5.144}
$$

The second map, represented by mean vector ${}^{\{2\}}\boldsymbol{\mu}_2$ and ${}^{\{2\}}\boldsymbol{\Sigma}_2$, where the left superscripts indicate as usual the map number and the right subscripts indicate the global time, is initialized with

- the robot position in the local origin ${}^{\{2\}}\boldsymbol{\Gamma}_2^{(2)}$ with zero variance and independent from other variables;
- the robot motion parameters ${}^{\{2\}}\boldsymbol{\Lambda}_2$ copied from ${}^{\{1\}}\boldsymbol{\Lambda}_2$;
- a second copy of the motion parameters ${}^{\{2\}}\boldsymbol{\Lambda}_s$ that represents the value of the parameters when the map was initialized;
- the landmark ${}^{\{2\}}\mathbf{y}_3{}^{(2)T}$ copied from ${}^{\{1\}}\mathbf{y}_3{}^{(2)T}$.

The new map results initialized as

$$
\begin{aligned}
{}^{\{2\}}\boldsymbol{\mu}_2 &= \left[{}^{\{2\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{2\}}\bar{\boldsymbol{\Lambda}}^{T},\ {}^{\{2\}}\bar{\boldsymbol{\Lambda}}_s{}^{T},\ {}^{\{2\}}\bar{\mathbf{y}}_3^{(2)T} \right]^T, \\
{}^{\{1\}}\boldsymbol{\Sigma}_2 &= \begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3} \\
\mathbf{0} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}_s}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\mathbf{y}_3} \\
\mathbf{0} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\mathbf{y}_3}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3}
\end{bmatrix}.
\end{aligned}
\tag{5.145}
$$

where the following equivalences are true

$$
\begin{aligned}
{}^{\{2\}}\bar{\boldsymbol{\Lambda}} &= {}^{\{1\}}\bar{\boldsymbol{\Lambda}}, \\
{}^{\{2\}}\bar{\boldsymbol{\Lambda}}_s &= {}^{\{1\}}\bar{\boldsymbol{\Lambda}}, \\
{}^{\{2\}}\bar{\mathbf{y}}_3^{(2)} &= {}^{\{1\}}\bar{\mathbf{y}}_3^{(2)}, \\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}} = {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}_s} &= {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}}, \\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3} = {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\mathbf{y}_3} &= {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3^{(2)}}, \\
{}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3} &= {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3^{(2)}\mathbf{y}_3^{(2)}}.
\end{aligned}
\tag{5.146}
$$

To complete the map transition step the map of correspondences SP is populated as

$$SP = \left\{ \begin{array}{ccccc} \langle & {}^{\{1\}}\mathbf{\Lambda} & , & {}^{\{2\}}\mathbf{\Lambda}_s & \rangle \\ \langle & {}^{\{1\}}\mathbf{y}_3{}^{(2)} & , & {}^{\{2\}}\mathbf{y}_3{}^{(2)} & \rangle \end{array} \right\}. \tag{5.147}$$

Notice that the ${}^{\{2\}}\mathbf{\Lambda}$ is copied from map 1, but it is not shared between map 1 and 2 since it will evolve during the robot motion. The shared element is ${}^{\{2\}}\mathbf{\Lambda}_s$, which represents the values of the motion parameters when map 2 was initialized.

Estimation goes on with the second map now. The landmark ${}^{\{2\}}\bar{\mathbf{y}}_3^{(2)}$ is observed and the new landmark ${}^{\{2\}}\bar{\mathbf{y}}_4^{(2)}$ is added to the map. Then a prediction step is performed, the landmark ${}^{\{2\}}\bar{\mathbf{y}}_4^{(2)}$ is observed and the new landmark ${}^{\{2\}}\bar{\mathbf{y}}_5^{(2)}$ is added to the map. The complete map results, at time 3, composed by

$$\begin{aligned}
{}^{\{2\}}\boldsymbol{\mu}_3 &= \left[{}^{\{2\}}\bar{\mathbf{\Gamma}}^{(2)T},\ {}^{\{2\}}\bar{\mathbf{\Lambda}}^T,\ {}^{\{2\}}\bar{\mathbf{\Lambda}}_s{}^T,\ {}^{\{2\}}\bar{\mathbf{y}}_3^{(2)T},\ {}^{\{2\}}\bar{\mathbf{y}}_4^{(2)T},\ {}^{\{2\}}\bar{\mathbf{y}}_5^{(2)T} \right]^T, \\
{}^{\{2\}}\mathbf{\Sigma}_2 &= \begin{bmatrix}
{}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{\Gamma}} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{\Lambda}} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{\Lambda}_s} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Gamma}\mathbf{y}_5} \\
{}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Gamma}\mathbf{\Lambda}} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}\mathbf{\Lambda}} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}\mathbf{\Lambda}_s} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}\mathbf{y}_5} \\
{}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Gamma}\mathbf{\Lambda}_s} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}\mathbf{\Lambda}_s} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}_s\mathbf{\Lambda}_s} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}_s\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}_s\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{\Lambda}_s\mathbf{y}_5} \\
{}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Gamma}\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}_s\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_3\mathbf{y}_3} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_3\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_3\mathbf{y}_5} \\
{}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Gamma}\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}_s\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{y}_3\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_4\mathbf{y}_4} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_4\mathbf{y}_5} \\
{}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Gamma}\mathbf{y}_5} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}\mathbf{y}_5} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{\Lambda}_s\mathbf{y}_5} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{y}_3\mathbf{y}_5} & {}^{\{2\}}\mathbf{\Sigma}^T_{\mathbf{y}_4\mathbf{y}_5} & {}^{\{2\}}\mathbf{\Sigma}_{\mathbf{y}_5\mathbf{y}_5}
\end{bmatrix}.
\end{aligned} \tag{5.148}$$

Let consider now to close a loop with ${}^{\{1\}}\mathbf{y}_1{}^{(1)}$. First of all, we have to perform the *back propagation* step from map 2 to map 1. According to the shared element map SP, we split the elements in map 2 in two groups:

$$\begin{aligned}
{}^{\{2\}}\mathbf{x}_B &= \left[{}^{\{2\}}\mathbf{\Gamma}^{(2)T},\ {}^{\{2\}}\mathbf{\Lambda}^T,\ {}^{\{2\}}\mathbf{y}_4^{(2)T},\ {}^{\{2\}}\mathbf{y}_5{}^{(2)T} \right]^T, && (5.149) \\
{}^{\{2\}}\mathbf{x}_C &= \left[{}^{\{2\}}\mathbf{\Lambda}_s{}^T,\ {}^{\{2\}}\mathbf{y}_3^{(2)T} \right]^T, && (5.150)
\end{aligned}$$

where ${}^{\{2\}}\mathbf{x}_C$ contains the elements that are shared between map 2 and 1, while ${}^{\{2\}}\mathbf{x}_B$ contains the elements that are only in map 2. The mean value vector and the covariance matrix can be partitioned as

$$\begin{aligned}
{}^{\{2\}}\boldsymbol{\mu}_3 &= \left[{}^{\{2\}}\boldsymbol{\mu}_B^T,\ {}^{\{2\}}\boldsymbol{\mu}_C^T \right] && (5.151) \\
{}^{\{2\}}\mathbf{\Sigma}_3 &= \begin{bmatrix} {}^{\{2\}}\mathbf{\Sigma}_{BB} & {}^{\{2\}}\mathbf{\Sigma}_{BC} \\ {}^{\{2\}}\mathbf{\Sigma}^T_{BC} & {}^{\{2\}}\mathbf{\Sigma}_{CC} \end{bmatrix}, && (5.152)
\end{aligned}$$

where

$$
\begin{aligned}
{}^{\{2\}}\boldsymbol{\mu}_B &= \left[{}^{\{2\}}\bar{\boldsymbol{\Gamma}}^{(2)T},\ {}^{\{2\}}\bar{\boldsymbol{\Lambda}}^T,\ {}^{\{2\}}\bar{\mathbf{y}}_4^{(2)T},\ {}^{\{2\}}\bar{\mathbf{y}}_5^{(2)T}\right]^T, && (5.153)\\
{}^{\{2\}}\boldsymbol{\mu}_C &= \left[{}^{\{2\}}\bar{\boldsymbol{\Lambda}}_s{}^T,\ {}^{\{2\}}\bar{\mathbf{y}}_3^{(2)T}\right]^T, && (5.154)\\
{}^{\{2\}}\boldsymbol{\Sigma}_{BB} &= \begin{bmatrix} {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_4} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_5}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_4} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_5}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_4}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_4}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_4\mathbf{y}_4} & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_4\mathbf{y}_5}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_5}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_5}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_4\mathbf{y}_5}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_5\mathbf{y}_5}\end{bmatrix}, && (5.155)\\
{}^{\{2\}}\boldsymbol{\Sigma}_{CC} &= \begin{bmatrix} {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\mathbf{y}_3}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}_s\mathbf{y}_3}^T & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3}\end{bmatrix}, && (5.156)\\
{}^{\{2\}}\boldsymbol{\Sigma}_{BC} &= \begin{bmatrix} {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_4\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_4\mathbf{y}_3}\\
{}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_5\boldsymbol{\Lambda}_s} & {}^{\{2\}}\boldsymbol{\Sigma}_{\mathbf{y}_5\mathbf{y}_3}\end{bmatrix}. && (5.157)
\end{aligned}
$$

Similarly, the map 1 has to be partitioned in two groups according to the shared elements map SP:

$$
\begin{aligned}
{}^{\{1\}}\mathbf{x}_A &= \left[{}^{\{1\}}\boldsymbol{\Gamma}^{(1)T},\ {}^{\{1\}}\mathbf{y}_1^{(1)T},\ {}^{\{1\}}\mathbf{y}_2{}^{(1)T},\ {}^{\{1\}}\mathbf{y}_3{}^{(1)T}\right]^T, && (5.158)\\
{}^{\{1\}}\mathbf{x}_C &= \left[{}^{\{1\}}\boldsymbol{\Lambda}^T,\ {}^{\{1\}}\mathbf{y}_3^{(2)T}\right]^T, && (5.159)
\end{aligned}
$$

where ${}^{\{1\}}\mathbf{x}_C$ contains the elements that are shared between map 1 and 2, while ${}^{\{1\}}\mathbf{x}_A$ contains the elements that are only in map 1. The mean value vector and the covariance matrix can be partitioned as

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_2 &= \left[{}^{\{1\}}\boldsymbol{\mu}_A^T,\ {}^{\{1\}}\boldsymbol{\mu}_C^T\right] && (5.160)\\
{}^{\{1\}}\boldsymbol{\Sigma}_2 &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{AA} & {}^{\{1\}}\boldsymbol{\Sigma}_{AC}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{AC}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{CC}\end{bmatrix}, && (5.161)
\end{aligned}
$$

where

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_A &= \left[{}^{\{1\}}\bar{\boldsymbol{\Gamma}}^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_1^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_2^{(1)T},\ {}^{\{1\}}\bar{\mathbf{y}}_3^{(1)T}\right]^T, & (5.162)\\
{}^{\{1\}}\boldsymbol{\mu}_C &= \left[{}^{\{1\}}\bar{\boldsymbol{\Lambda}}^{T},\ {}^{\{1\}}\bar{\mathbf{y}}_3^{(2)T}\right]^T, & (5.163)\\
{}^{\{1\}}\boldsymbol{\Sigma}_{AA} &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_1}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_2}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_2}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_2} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3} \end{bmatrix}, & (5.164)\\
{}^{\{1\}}\boldsymbol{\Sigma}_{CC} &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3^{(2)}}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Lambda}\mathbf{y}_3^{(2)}}^T & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3^{(2)}\mathbf{y}_3^{(2)}} \end{bmatrix}, & (5.165)\\
{}^{\{1\}}\boldsymbol{\Sigma}_{AC} &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}\mathbf{y}_3^{(2)}}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_3^{(2)}}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_2\mathbf{y}_3^{(2)}}\\ {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\boldsymbol{\Lambda}} & {}^{\{1\}}\boldsymbol{\Sigma}_{\mathbf{y}_3\mathbf{y}_3^{(2)}} \end{bmatrix}. & (5.166)
\end{aligned}
$$

The back propagation equations (Equations 5.106 to 5.109) can be applied and the map 1 results updated with the contributions of the second map observations as

$$
\begin{aligned}
{}^{\{1\}}\boldsymbol{\mu}_2^* &= \left[{}^{\{1\}}\boldsymbol{\mu}_A^{*\,T},\ {}^{\{2\}}\boldsymbol{\mu}_C^T\right] & (5.167)\\
{}^{\{1\}}\boldsymbol{\Sigma}_2^* &= \begin{bmatrix} {}^{\{1\}}\boldsymbol{\Sigma}_{AA}^* & {}^{\{1\}}\boldsymbol{\Sigma}_{AC}^*\\ {}^{\{1\}}\boldsymbol{\Sigma}_{AC}^{*\,T} & {}^{\{2\}}\boldsymbol{\Sigma}_{CC} \end{bmatrix}. & (5.168)
\end{aligned}
$$

To perform the loop closure with the landmark ${}^{\{1\}}\mathbf{y}_1{}^{(1)}$ we have to follow the next steps

- transform ${}^{\{1\}}\mathbf{y}_1{}^{(1)}$ to ${}^{\{1\}}\mathbf{y}_1{}^{(2)}$ with Equation 5.86;
- add ${}^{\{2\}}\mathbf{y}_1{}^{(2)}$ to the second map, coping it from ${}^{\{1\}}\mathbf{y}_1{}^{(2)}$, in particular:
  - the mean values ${}^{\{2\}}\bar{\mathbf{y}}_1^{(2)}$ are copied from ${}^{\{1\}}\bar{\mathbf{y}}_1^{(2)}$,
  - covariances between the new landmark ${}^{\{2\}}\bar{\mathbf{y}}_1^{(2)}$ and the shared elements ${}^{\{2\}}\mathbf{x}_C$ are copied from the covariances in the original map, i.e., covariances between ${}^{\{1\}}\bar{\mathbf{y}}_1^{(2)}$ ${}^{\{1\}}\mathbf{x}_C$,
  - covariances between the new landmark ${}^{\{2\}}\bar{\mathbf{y}}_1^{(2)}$ and the elements of the second map ${}^{\{2\}}\mathbf{x}_B$ are calculated with Equation 5.137;
- the pair $\left\langle {}^{\{1\}}\mathbf{y}_1{}^{(2)}, {}^{\{2\}}\mathbf{y}_1{}^{(2)} \right\rangle$ is added to the SP map of the shared elements.

Part II

Large Scale Real Time Visual SLAM

CHAPTER 6

Mono and Multi Camera EKF Visual SLAM

The generic EKF SLAM presented in Section 5.4 has been successfully applied to the Visual SLAM problem, i.e., to a particular SLAM problem statement that rely on measurements provided by images perceived by cameras. This chapter provides all the details that allow to implement a complete EKF Visual SLAM system starting from the generic description of the algorithms.

6.1 Introduction

Visual SLAM, i.e., a specific application of SLAM that uses cameras as sensors for the landmark perception and measurement, is considered a very promising application of SLAM, since cameras are massively diffuse, cheap, low-power and lightweight. Moreover, cameras image streams provide a massive amount of information, since a thousand pixels wide image is gathered in a fraction of a second. Visual SLAM has gained interest in field different from the pure robotics area; for instance, visual SLAM can be used to build the map of an environment in which a person is moving freely while holding a smart-phone in hand.

Visual SLAM takes the names of *Monocular SLAM* when only a single camera is used, *Stereo SLAM* when two cameras are used , *Trinocular SLAM* when the cameras are three and *Multi Camera SLAM* when a generic number of cameras is used. One of the first works implementing an on-line stereo SLAM working in real time is [21]. It used the Shi-Tomasi salient point detector and small patches as feature descriptors matched by correlation. One of the key aspect presented in this work is the *active search* approach to measurements:

the EKF-SLAM mechanism allows to predict the projection of landmarks in the current image, thanks to the measurement step. By computing the covariance associated with the measure we obtain a 2D elliptical region, in terms of Mahalanobis distance, around the mean prediction, which bounds the search area in the image that contains the feature with some given probability. This approach reduce the research area, cutting down the computational cost and data association errors.

The system presented in [20] was the first on-line SLAM system working with a single camera, i.e., the first monocular SLAM system. The monocular approach is much more challenging than the stereo in several ways. First of all, the proposed approach does not use any odometric information, since the camera is held in the hand of a human operator. Secondly, a monocular camera is a bearing only sensor; it can perceive only the direction of the viewing ray of a projected environmental point, but it can not know the distance. The proposed solution uses a *delayed approach*, i.e., the landmark are added as 3D point to the filter once a proper estimation of the depth, performed by a separate particle filter, has been reached. Consequently, the system needs to be started with a visible known pattern, in order to track the first movement up to the estimation of the depth of some new landmarks. This work is the seminal paper that gave birth to an important research effort. Although it was the first running EKF-SLAM system, most of its concepts and techniques are still a valid approach and the EKF-SLAM system presented in this work is highly based on the original proposal.

The first attempt to abandon the delayed initialization, i.e., to introduce landmarks in the filter state and use them in the measurement process from the first time they have been seen, was developed in [89] and [92], with the *Federate Information Sharing (FIS)* approach. FIS is an approximation of the Gaussian Sum Filter which introduces landmarks as a sum of Gaussian hypothesis along the viewing ray of the landmark. Once the landmark starts to be measured, the hypothesis are weighted by their likelihood and the less probable are pruned after a few steps.

The delayed initialization was definitely abandoned with the introduction of the Unified Inverse Depth Parameterization (UID), proposed in [65]. This work introduces the concept of *parameterization*, i.e., the description of a 3D point representing a landmark with a different formulation that allows to take in account the initial uncertainty on the depth of the perceived landmark. The UID parameterization was successfully applied in numerous works, like [75], [67], [100]. Since then, other parameterizations have been introduced, like the Inverse Scaling (IS) [62] [63], the Anchored Homogeneous Point (AHP) [90] [93]. A good review of these parameterizations, among with an experimental evaluation of their properties, can be found in [90] and [93]. Here we present two different parameterizations, the Framed Homogeneous Point (FHP), originally presented in [10], and the Framed Inverse Scale (FIS). These parameterizations are not completely new: a similar version of FHP can be found in [43], while a proposal similar to FIS is in [74]. Both well-known parameterizations and the new proposals are presented here in a different form respect the original formulation. Such a reformulation allows to point out a state vector size saving, thus a computational complexity reduction, and to solve some issues related to their use of the CI-SLAM framework.

In [91] an interesting discussion about the usage of the monocular approach to multi-camera systems was proposed with the provocative title “BiCamSLAM: Two times mono is more than stereo”. In standard approaches, stereo cameras and multi camera systems are used taking into account the underlying mathematical and geometrical concepts (e.g., the epipolar geometry) to initialize 3D landmarks by triangulation of corresponding features. The *BiCamSlam* approach motivates precisely the benefits of the implicit use of a multi camera system. Basically, the proposal relies on the usage of a monocular approach for the landmark addition and to perform measurements in all the cameras of the multi camera system, allowing the use of an active search approach for template matching. Triangulation is never performed explicitly, but landmarks that are matched in more than one camera generates measurement equations that updates properly the depth estimation. The author lists the next reasons for the usage of a *BiCamSlam* approach:

- Depth of points that are close to the camera are rapidly estimated thanks to the double, in a stereo setup, or multiple, in a generic setup, measurements.
- Bearing only measurements (i.e., measures from a single camera) allow the estimation of the orientation, especially for the very far landmarks
- Updates can be performed on any landmark that is only visible from one camera.

Moreover, the author indicates that the precise calibration of the relative camera position and orientation is no longer necessary and it can be performed in the SLAM algorithm directly.

6.2 Multi Camera EKF-SLAM With Conditional Independance Submapping

In the remaining of this chapter the algorithm for a multi camera EKF-SLAM system, based on the *BiCamSlam* approach and fused in the CI-SLAM framework, will be detailed. In this section, the general algorithm is depicted and its differences with the classical EKF-SLAM system presented in Section 5.4 are pointed out. In the next sections the fundamental mathematical components, i.e., the motion models equations and the parameterizations of the landmarks, of the system are detailed.

6.2.1 The main loop

The main loop of the multi camera EKF-SLAM system has to prepare the first map, take images from the streams, measure the landmarks in each camera, i.e., predict their position in the images and search for them in the expected images regions, update the EKF with the measurements and manage the map in order to add new landmarks if needed or to delete unuseful landmarks. Moreover, the main loop has to check if the current map needs to be closed. In this case it has to trigger a map transition procedure. As a last step, the prediction of the EKF takes place and the entire loop is repeated for the next frame.

Let consider Algorithm 7; in steps $2-3$ the first map is initialized and added to the set of maps M. The loop from line 4 to 31 iterate the multi camera SLAM system operations for each frame. In particular, for each camera of the system (step 6) a new image is acquired (step 7) and the landmarks in the current map m_i are projected in the image (step 8) according to proper measurement equations. The projection allows to start an active matching search on the ellipsoidal areas in which the feature is expected to be (step

Algorithm 7 Main Loop

```
1: # Initialization
2: initialize map $m_1$
3: $M \leftarrow m_1, i = 1$
4: loop
5:    # Measurement step
6:    for all $c$ in cameras $C$ do
7:       acquire image $\mathbf{I}_c$ from the stream
8:       evaluate measurement predictions in image $\mathbf{I}_c$
9:       find correspondences through active search mechanism
10:      collect individual compatible measures
11:   end for
12:   # Update step
13:   perform EKF update step
14:   # CI-SLAM maps management
15:   if map $m_i$ has to be closed then
16:      perform map transition step, initialize map $m_{i+1}$
17:      $M \leftarrow \{M, m_{i+1}\}, i \leftarrow i + 1$
18:   end if
19:   # Current map management
20:   for all $c$ in $C_I$ do
21:      search for new landmarks in $\mathbf{I}_c$
22:      add landmarks to the map
23:      for all $d$ in $C \backslash C_I$ do
24:         evaluate measurement predictions in image $\mathbf{I}_d$
25:         find correspondences through active search mechanism
26:         collect individual compatible measures
27:      end for
28:   end for
29:   if $\exists$ matched new landmark then
30:      perform EKF update
31:   end if
32:   delete unstable features
33:   # Prediction Step
34:   perform EKF prediction step
35: end loop
```

9). The matched features are collected in an unique set of the individually compatible measurements (step 10). After all the cameras are processed the update of the EKF state can be performed with the complete set of collected measurements. This step is skipped if no measurements are available, alike at the first step of the execution, where no landmarks are in the map. After the map is properly updated a condition on the map closure is verified (step 15) and a new map is stared if necessary with a map transition procedure (step 16-17). Right before the motion prediction step, the current map is analyzed and processed (steps 20 to 28) in order to add new landmarks if needed and to delete landmarks that are not useful anymore.

Landmark addition is performed on a subset C_I of all the cameras C. For instance, in a stereo camera setup the left camera can be used to initialize new landmarks while the right one only for the measurements. For each camera enabled for the landmark initialization (step 20), new landmarks are searched in the current image (step 21), then they are added to the map, i.e., to the EKF (step 22). Subsequently the new landmarks are projected in all the cameras beside the one in which they have been initialized and then they are matched through the active search mechanism (steps 23 to 27). If some new landmark has been properly matched an update of the EKF is performed. Consequently, new landmarks that have been observed by more than one camera have a properly updated depth, while others new landmarks are simply added to the map as in the monocular case. The last step of the map management searches for unuseful landmarks and deletes them from the map (step 32). Landmarks are considered useless, and potentially harmful, when they are predicted to be in the image for a certain amount of time, but they can not be matched. Before we restart the main loop, the motion prediction step is performed (step 34) according to the characteristic of the application (i.e., using odometric information or a constant velocity motion model).

It has to be noticed that the backpropagation step of the CI-SLAM framework is not described in the main loop since it can be executed at any time in a parallel thread. In a sequential implementation it can be executed when a map is closed (after step 15) or alternatively postponed to the end of the SLAM execution to refine the overall estimate. Similarly, for the loop closure procedure of the CI-SLAM framework, we have to imagine that in a complete SLAM implementation, a dedicated procedure for the *loop detection* is executed in parallel to the SLAM system to trig the execution of a loop closure procedure. We will not treat the loop detection problem in this work, the interested reader can refers to the huge literature, starting from the comparison of loop detection methods proposed in [13] and exploring the state of art methods, such as the works presented in [31], [8], [32], [60], [70], [69], [59], [42].

The proposed form of the algorithm for the multi camera SLAM system assumes that cameras are triggered and they produce images synchronously. If cameras have different frame rates or the acquisition is not synchronized we can modify the algorithm in order to process the images at the right time, hypothesizing that they are provided with a precise timestamp and the same base time. Moreover, the algorithm can be easily adapted to an *active SLAM* application, i.e., to an application in which cameras are inserted and removed from the estimation process at run time. Such an algorithm can be useful, for instance, when low power consumption is requested and camera could be easily turned on and off when needed.

6.2.2 Map management strategies

In this section we describe with more details some map management strategies, i.e., policies for landmark addition and deletion, that we have applied in our work and experiments. Some of them are suitable for real experiments while some others are applicable only in simulation.

6.2.2.1 Landmark Addition Strategies

We have used three different landmarks addition strategies, we name them *grid-based* strategy, *randomized* strategy and *perfect-knowledge* strategy. The first two of them work with real images while the third one only in simulation. The goal of a landmark addition strategy is to guarantee the presence of a sufficient number of landmarks in the image at each iteration, ensuring a good distribution of points over the entire image. Landmark addition uses a salient point detector to identify the locations of the interesting features that describe the landmarks.

The *grid-based* strategy subdivides the image in a matrix of $n_C = r \times c$ equal size cells. The policy looks for new landmarks only if in the current image less than a fixed threshold n_I landmarks are visible, otherwise the addition is skipped. Salient point detector is applied to the cells which have a number of visible landmark lower than $l_C = \frac{n_I}{n_C}$. The detection of point where it is needed allows to reduce computational requirements, since non interesting zoneS are not processed. Moreover, the salient point detector is instructed to mask the elliptical areas where the visible landmarks are, avoiding landmark duplication or landmarks too close to existing ones. The maximum number of landmarks that can be added at the same time is specified by a threshold max_{new}, but in the first frame, were no landmarks are in the image, a different threshold max_0 is used to initialize the map properly.

The *randomized* strategy tries to find “empty” regions in the image and looks for salient points in them. A image coordinate $\boldsymbol{p}$ is randomly generated and the number of visible landmarks in a surrounding area of $w \times h$ is computed. If no landmarks are in the area the salient point detector is applied to the area to detected corners (if any), i.e., new landmark candidates. Then, the procedure is repeated until a maximum number of attempts is reached or the maximum number of landmarks visible in the image n_I is reached. In practical applications we observe substantially equivalent performances between the two policies, although a deep investigation on this was not carried out.

The *perfect-knowledge* strategy is a special policy that can be applied only in simulation. With the term “simulation” we mean that no real image is provided, but the coordinates of a set of 3D points coming from a known map are projected in the image knowing the real robot (or camera) position. This results in a set of 2D points that represent the landmarks in the images. In this context, we assume to known perfectly the data association, i.e., point are described by a unique-id. The *perfect-knowledge* landmark addition policy leverages on the a-priori knowledge and selects new landmarks if they are visible in the current image and they are not in the filter state.

6.2.2.2 Landmark Deletion Strategies

The landmark deletion step aims at purging from the map, then from the filter, the landmarks that are no more useful to the estimation process or possibly harmful. In particular, each landmark is individually tested with a series of boolean predicates that vote for its deletion. The results of these tests are combined with Boolean algebra operator and the final result represents the choice on the deletion. We individuate the following predicates, where the *true* value correspond to a proposal for deletion:

- *Lifetime*: a landmark that has been added to the filter more than min_t steps ago can be deleted;
- *Match Ratio*: a landmark with match ratio α less than min_α can be deleted, where α is the ration between the number of frame in which the landmark is predicted to be visible and the number of frames in which the landmark has been matched;
- *Broken*: a landmark is said to be *broken* if it has some characteristic that makes it unuseful. Although it can not be completely clear at this point of the explanation, for the landmark representation we use a landmark is said to be broken if its *depth* is negative.

The deletion strategy we adopt is the following:

$$(\mathrm{Lifetime} \wedge \mathrm{MatchRatio}) \vee \mathrm{Broken}, \tag{6.1}$$

i.e., a landmark is deleted if it is said to be broken or it has matched a low number of times (it is somehow *unstable*), although it has been in the filter for a sufficient time. Without this last condition we risk to delete landmarks too frequently. Consider, for instance a match ratio $\alpha = 0.5$. If a landmark is added and then it is not matched in the next frame, it would be deleted, preventing a possible match in a following frame. The introduction of the *Lifetime* predicate add robustness to this test guaranteeing a minimum number of attempts to the matcher.

A special attention as to be paid in the deletion of landmarks that are shared with a previous map in the CI SLAM framework. We can adopt two strategies: the first is to remove landmarks from the current map and from the other maps where it is shared; the second is to avoid the deletion of such landmarks. We adopt the latter policy to ease the implementation.

The definition of predicates for landmark selection is particularly useful since, with simple modifications, the SLAM algorithm can be converted in a visual Odometry system, which is particularly useful for testing and debugging. If we consider the deletion strategy

$$(\mathrm{Lifetime} \wedge \mathrm{MatchRatio}) \vee \mathrm{Broken} \vee (\mathrm{NotInImage} \wedge \mathrm{Lifetime})\,, \tag{6.2}$$

where $\mathrm{NotInImage}$ is true if the landmark is predicted to be outside of the image, old landmarks that are no more visible are deleted, thus the map is forgotten, limiting the number of landmarks in the filter and allowing a very fast execution.

As a final remark, we do not invoke the landmarks deletion procedure at each iteration of the SLAM system, but only periodically. This is because deletion of landmarks requires to resize the filter state and to reorder elements in the mean vector and in the covariance matrix. Doing this periodically for a set of landmarks save computation time.

6.3 Motion Models

Let start the detailed description of the multi camera SLAM system components from the so called *motion model*, i.e., the state transition probability function that drives the prediction step of the EKF SLAM algorithm. With reference to Section 5.4.3, and in particular to Equation 5.8, we have to specify the function $f_{\mathbf{D}}(\cdots)$. Here we presents three different formulations, one of them needs odometric information as input, while the remaining two assume a constant velocity motion model and they do not require any input apart from the previous state estimation.

Lets remind the form of the state transition function of Equation 5.8,

$$\left[\boldsymbol{\Gamma}_k^T,\, \boldsymbol{\Lambda}_k^T\right]^T \quad = \quad f_{\mathbf{D}}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{6.3}$$

where

- $\boldsymbol{\Gamma}_{k-1}$ and $\boldsymbol{\Gamma}_k$ are respectively the estimation of the current robot pose and the prediction of the next robot pose. From now on $\boldsymbol{\Gamma}$ is considered as a 7-element variable $\left[\mathbf{t}_k^T = \mathbf{t}_{\mathcal{R}}^{\mathcal{W}^T},\, \mathbf{q}_k^T = \mathbf{q}_{\mathcal{R}}^{\mathcal{W}^T}\right]$, composed by a translation vector and a quaternion, representing the pose of the robot (R) with respect to the global reference frame (W), i.e., the transformation $\mathbf{T}_{\mathcal{R}}^{\mathcal{W}}$. Different representations of the robot pose are possible, in particular for the representation of the rotation, but hereafter we consider the quaternions;

- $\boldsymbol{\Lambda}_{k-1}$ and $\boldsymbol{\Lambda}_k$ represent the motion parameters. As already explained in Section 5.4.1 this term can be omitted in some cases. In particular, we do not make use of it when odometric information is provided; in that case only the robot pose is part of the estimation. $\boldsymbol{\Lambda}$ is used to represent the robot tangential and rotational velocity when odometric information is not available. Tangential and rotational velocity are expressed with two 3-element vector $\mathbf{v}$ and $\boldsymbol{\omega}$. In particular, tangential velocity, coded in a 3-element vector, can be expressed with respect to the robot reference frame R or with respect to the word reference frame W, depending on the specific model. Rotational velocity is expressed with a rotation vector in the unit of time referred to the robot reference system;

- $\mathbf{u}_k$ represents the control input and in our case we use odometric information. In particular we consider a complete 6 degree of freedom model, although the odometric information might interest only a subset of them. The odometric input is represented by a 6-element vector representing the displacement from the last robot pose in terms of a translation vector $\boldsymbol{\Delta}\mathbf{t}_k$ and a rotation vector $\boldsymbol{\Delta}\mathbf{r}_k$. These values are expressed in the robot reference system, since they are perceived by on-board sensors. If the odometric information are not available, this term is omitted;

- $\boldsymbol{\eta}_k$ is a zero mean Gaussian noise that models the uncertainty. In the motion model with odometric information noise is generally considered additive to the input, while in constant velocity motion models noise refers the unmodeled elements, e.g., accelerations. Alike for other elements, it is convenient to consider the 6-elements noise vector as partitioned in two vectors. In particular, we use $\boldsymbol{\eta}_{\mathbf{t}k}$ and $\boldsymbol{\eta}_{\mathbf{r}k}$ to represent noise respectively on the translation part and on the rotation part when odometric information is available; we use $\boldsymbol{\eta}_{\mathbf{v}k}$ and $\boldsymbol{\eta}_{\boldsymbol{\omega}k}$ to represent noise in motion models without odometric information.

6.3.1 Motion Model With Odometric Information

When odometric information is available, the element $\boldsymbol{\Lambda}$ is omitted from the state vector and the complete motion model equations is

$$\boldsymbol{\Gamma}_k = f_{\mathbf{D}}^{ODO}(\boldsymbol{\Gamma}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{6.4}$$

$$\begin{bmatrix} \mathbf{t}_k \\ \mathbf{q}_k \end{bmatrix} = \begin{bmatrix} \mathbf{t}_{k-1} + \mathbf{R}\left(\mathbf{q}_{k-1}\right)\left(\boldsymbol{\Delta}\mathbf{t}_k + \boldsymbol{\eta}_{\mathbf{t}k}\right) \\ \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\Delta}\mathbf{r}_k + \boldsymbol{\eta}_{\mathbf{r}k} \end{bmatrix} \end{bmatrix}. \tag{6.5}$$

These equations come from transformation composition rules: $\boldsymbol{\Gamma}_{k-1}$ is the robot position at previous step with respect to the world reference frame, $\mathbf{u}_k$ represent the new pose of the robot at step k expressed in the reference system of the robot at time $k-1$. Under the hypothesis of small rotations it is possible to compute the quaternion update equation from the time derivatives presented in Section 2.3.6.

Beside the motion function definition, we need to compute the Jacobians of the $f_{\mathbf{D}}^{ODO}(\cdots)$ function as

- $\frac{\partial f_{\mathbf{D}}^{ODO}(\cdots)}{\partial \boldsymbol{\Gamma}_{k-1}}$, i.e., derivatives with respect to the dynamic part of the state vector,
- $\frac{\partial f_{\mathbf{D}}^{ODO}(\cdots)}{\partial \boldsymbol{\eta}_k}$, i.e., derivatives with respect to the noise of the model.

It can be noticed that in this model $\frac{\partial f_{\mathbf{D}}^{ODO}(\cdots)}{\partial \boldsymbol{\eta}_k} = \frac{\partial f_{\mathbf{D}}^{ODO}(\cdots)}{\partial \mathbf{u}_k}$, since noises are additive to the inputs.

Rotations are represented by quaternions with a unitary module. This is true at the initialization time, where the $\boldsymbol{\Gamma}_0$ element in the EKF is initialized as the origin ($\boldsymbol{\Gamma}_0 = [0,0,0,1,0,0,0]^T$) with zero uncertainty ($\boldsymbol{\Sigma}_0 = \mathbf{0}_{7\times7}$) but the motion equation and the update step of the EKF may result in a non unitary quaternion: $\|\mathbf{q}_k\| \neq 1$. This problem can be addressed with a quaternion normalization step that modifies the rotation variable and imposes the unity of the module

$$\mathbf{q}_k = \frac{\tilde{\mathbf{q}}_k}{\|\tilde{\mathbf{q}}_k\|}, \tag{6.6}$$

where the tilde symbol indicates the rotation variable before normalization. This step has to modify the covariance matrix of the filter through Jacobian of the normalization function. Since the only involved variable is the rotation itself, it can be shown that the elements modified in the covariance matrix are the vertical and horizontal band of the covariance matrix corresponding to the variable position only. This implies that the normalization step has the same complexity of the prediction step, i.e., $O(n)$ being n the number of landmarks in the filter.

The noise in the odometry is usually assumed to be independent, thus the covariance matrix is a diagonal matrix. Empiric considerations and some evidence from experiments shows that the more is the robot motion, the more is the uncertainty on the error. Thus, the non zero elements $\mathbf{\Sigma}_{ii}$ of the covariance matrix are calculated following the linear rule

$$\mathbf{\Sigma}_{ii} \quad = \quad \left(\boldsymbol{\sigma}_{0i} + s_i \mathbf{u}_i\right)^2 , \tag{6.7}$$

where $\boldsymbol{\sigma}_{0i}$ is some minimum standard deviation, used when there is no motion on the i value of the input vector and s_i is a scale factor that increments linearly the standard deviation with the input $\mathbf{u}_i$.

6.3.2 Motion Models Without Odometric Informations

When odometric information is not available, e.g., in the case of pure monocular camera or with a stereo camera in hand, it is possible to consider a constant velocity motion model as a good approximation of the real motion. In this case the prediction of the next pose is based on the current robot velocity, represented in the element $\mathbf{\Lambda}$ which is part of the state vector. Under the hypothesis that the motion is sufficiently smooth, the prediction of the velocity maintains the same velocity but enlarge its variance to take in account changes that may happens in the reality, i.e., to model the accelerations.

If no additional information on the initial motion is available, the initial mean values of the $\mathbf{\Lambda}$ Gaussian variable are all zeros, assuming the robot is initially steady with null speed, but, to be general, with covariance matrix $\mathbf{\Sigma}_{\mathbf{\Lambda}}$ different from zero. This results in

$$\boldsymbol{\mu}_0 = \begin{bmatrix} [0,0,0,1,0,0,0]^T \\ [0,0,0,0,0,0]^T \end{bmatrix} . \tag{6.8}$$

The covariance associated with the initial robot pose is set to zero, so the initial covariance matrix results

$$\mathbf{\Sigma}_0 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{\mathbf{\Lambda}} \end{bmatrix} . \tag{6.9}$$

When the element $\mathbf{\Lambda}_k$ is composed by the tangential velocity $\mathbf{v}_k^{(\mathcal{W})}$ expressed in world reference frame and the rotational velocity vector $\boldsymbol{\omega}_k$ in robot reference frame the motion model results in

$$\begin{bmatrix} \mathbf{\Gamma}_k \\ \mathbf{\Lambda}_k \end{bmatrix} \quad = \quad f_{\mathbf{D}}^{CSW}(\mathbf{\Gamma}_{k-1}, \mathbf{\Lambda}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{6.10}$$

$$\begin{bmatrix} \mathbf{t}_k \\ \mathbf{q}_k \\ \mathbf{v}_k^{(\mathcal{W})} \\ \boldsymbol{\omega}_k \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{t}_{k-1} + \left(\mathbf{v}_k^{(\mathcal{W})} + \boldsymbol{\eta}_{\mathbf{v}k}\right) \Delta t \\ \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix} 0 \\ \left(\boldsymbol{\omega}_k + \boldsymbol{\eta}_{\boldsymbol{\omega}k}\right) \Delta t \end{bmatrix} \\ \mathbf{v}_{k-1}^{(\mathcal{W})} + \boldsymbol{\eta}_{\mathbf{v}k} \\ \boldsymbol{\omega}_{k-1} + \boldsymbol{\eta}_{\boldsymbol{\omega}k} \end{bmatrix} , \tag{6.11}$$

where Δt is the time of a discrete step. If the tangential velocity is expressed in the robot reference frame $\mathbf{v}_k^{(\mathcal{R})}$ the equation changes as

$$\begin{bmatrix}\mathbf{\Gamma}_k \\ \mathbf{\Lambda}_k\end{bmatrix} = f_{\mathbf{D}}^{CSR}(\mathbf{\Gamma}_{k-1}, \mathbf{\Lambda}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k) \tag{6.12}$$

$$\begin{bmatrix}\mathbf{t}_k \\ \mathbf{q}_k \\ \mathbf{v}_k^{(\mathcal{R})} \\ \boldsymbol{\omega}_k\end{bmatrix} = \begin{bmatrix}\mathbf{t}_{k-1} + \mathbf{R}\left(\mathbf{q}_{k-1}\right)\left(\mathbf{v}_k^{(\mathcal{R})} + \boldsymbol{\eta}_{\mathbf{v}k}\right)\Delta t \\ \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix}0 \\ \left(\boldsymbol{\omega}_k + \boldsymbol{\eta}_{\boldsymbol{\omega}k}\right)\Delta t\end{bmatrix} \\ \mathbf{v}_{k-1}^{(\mathcal{R})} + \boldsymbol{\eta}_{\mathbf{v}k} \\ \boldsymbol{\omega}_{k-1} + \boldsymbol{\eta}_{\boldsymbol{\omega}k}\end{bmatrix} \tag{6.13}$$

As stated in the previous section, notice that the quaternion element needs to be normalized after the prediction step in order to ensure that its module is maintained unitary.

The second model (Equation 6.13) is more suitable when some additional information on the robot motion is available. Lets consider for instance a wheeled holonomic robot moving on the floor without odometric information. We can assume that the motion has a “preferred” direction, since the robot can only translate in the forward direction and rotate around the vertical axis, while it can not fly, thus changes in height are necessarily smooth and due to changes in terrain slope. At the same manner, rotation around axis parallel to the ground are quite impossible, and small variations are due to roughness of the ground. Noises $\boldsymbol{\eta}_k$ expressed in robot reference frame can be used to properly model these physical constraints.

6.3.2.1 Use in the CI Slam framework

As stated in the Section 5.6.2.2, the $\mathbf{\Lambda}$ variable has to be introduced in the shared variables when a new map is created. In particular, in the case of the motion model of Equation 6.10, this variable needs to be transformed in the new reference system, being the tangential velocity referred to the world reference frame (i.e., to the current map reference frame). The transformation function $pt(\cdots)$ of Equation 5.95, expressed only for the tangential part, is

$$\mathbf{v}_k^{(i+1)} = \mathbf{R}\left(\mathbf{q}_k\right)^T \mathbf{v}_k^{(i)}, \tag{6.14}$$

i.e., the tangential velocity expressed in the reference system of the actual map i is rotated in the current robot reference frame by the inverse rotation expressed by the quaternion of current the robot orientation.

6.4 Parameterizations

Lets now consider the landmarks and the specific functions that, relying on the current robot position and on the acquired measure, allow to measure a landmark that is already in the state vector or to a add a new landmark. Moreover, the use in the CI SLAM framework needs landmarks, referred to the map reference frame, to be transformed in the new reference frame to be shared with the new map.

Up to now it was not specified precisely what a landmark is. For the purpose of this work we consider a landmark as a 3D Euclidean point in the environment. However, 3D points are not suitable to be used directly in the state vector to code landmarks when a single camera (monocular) visual SLAM system is considered, as stated in the introduction of this chapter. Different *parametrization*, i.e., description of a 3D point through a different formulation, have been developed in the literature, while two new proposal are presented here. It has to be noticed that we present here well known literature parameterizations in a slightly different way, providing some considerations that so far remained hidden in the main presentations done in the literature.

To be general, we assume the camera (or the cameras) to be placed in a different reference frame with respect to the robot reference frame. Lets name $\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}$ the pose of the i-th camera with respect to the robot reference system expressed with a translation vector $\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}$ and a quaternion $\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}}$ for the orientation. These transformations can be obtained with an extrinsic calibration of the cameras and they are used in the landmarks addition and measurement equation.

6.4.1 Definitions

In this section some terms are clearly defined and then used in the rest of the chapter to define the possible parameterizations.

6.4.1.1 Landmark Addition

First of all we consider the steps that lead to the addition of a new landmark to the filter state. In the n-cameras SLAM setup described in the introduction of this chapter we have to consider that a landmark is perceived by a single camera as a point in the current image. The camera geometry equations allow to compute the direction of its corresponding viewing ray, while from a single image it is not possible to known the distance of the observed point.

Considering the transformation that converts an image point $\boldsymbol{p}_{img}$ in a direction vector, lets recall the inverse of the projection equation of Section 3.1.7, expressed through function $K^{-1}(\cdots)$, an name $\mathbf{r}^{(\mathcal{C})}$ the viewing ray exiting from the camera reference frame when the image point $\boldsymbol{p}_{img}$ is considered:

$$\mathbf{r}^{(\mathcal{C})} = K(\boldsymbol{p}_{img})^{-1}. \tag{6.15}$$

The coordinate of the point $\boldsymbol{p}_{img}$ may be affected by some noise coming from many sources, e.g., imprecise calibrations or inaccuracy of the salient point detection mechanism. To accommodate these situations we define $\tilde{\boldsymbol{p}}_{img}$ as the true image point $\boldsymbol{p}_{img}$ affected by an additive zero mean Gaussian noise $\boldsymbol{\xi}_{\boldsymbol{p}_{img}}$, resulting in

$$\tilde{\boldsymbol{p}}_{img} = \boldsymbol{p}_{img} + \boldsymbol{\xi}_{\boldsymbol{p}_{img}}, \tag{6.16}$$

The noise on the two dimension is generally considered uncorrelated, thus the covariance matrix of $\boldsymbol{\xi}_{\boldsymbol{p}_{img}}$ is diagonal. A common assumption is to consider a standard deviation of 1 pixel for the noise, thus the covariance matrix results in the identity $\boldsymbol{\Sigma}_{\boldsymbol{\xi}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

The vector $\mathbf{r}^{(\mathcal{C})}$ applied to the camera center reach the normalized image plane, i.e., it has the z coordinate equal to 1, thus the first two element of it represent the 2D coordinate of point $\boldsymbol{p}_{\pi}$ corresponding to $\tilde{\boldsymbol{p}}_{img}$ on the normalized image plane, defined also as $\boldsymbol{p}_{\pi} = K^{I}(\boldsymbol{p}_{img})^{-1}$. Similarly we can consider the image point affected by noise, obtaining $\tilde{\boldsymbol{p}}_{\pi} = K^{I}(\tilde{\boldsymbol{p}}_{img})^{-1}$.

The unit vector $\bar{\mathbf{r}}^{(\mathcal{C})}$ can be defined as the ratio between the viewing ray vector and its module

$$\bar{\mathbf{r}}^{(\mathcal{C})} = \frac{\mathbf{r}^{(\mathcal{C})}}{\|\mathbf{r}^{(\mathcal{C})}\|}, \tag{6.17}$$

while, considering noise sources, we define the unit vector $\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}$ as

$$\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})} = \frac{\tilde{\mathbf{r}}^{(\mathcal{C})}}{\|\mathbf{r}^{(\mathcal{C})}\|}. \tag{6.18}$$

As we stated before, from a single image it is not possible to know the distance of a 3D point projected on the image, since all points that lie on the viewing ray have the same projection. This implies that the distance of the observed point from the camera center is uniformly distributed from zero to the infinity. In the literature it was clearly demonstrated, and it is intuitively clear, that such unknown distance can not be properly approximated by a Gaussian distribution. In fact, even if the uncertainty is set to a very big value, the infinite can not be never included. From this consideration it follows that an Euclidean 3D point with a Gaussian uncertainty distribution is not a suitable solution. The description of the depth d in terms of its inverse, i.e., $d = 1/\varrho$ allows to include the infinity ($\varrho = 0$) and introduces a good approximation of the theoretic uniform distribution. When the landmark is firstly perceived, its initial depth is unknown, thus its inverse depth can be specified with an initial value ϱ_0 and an initial associated variance $\sigma^2_{\varrho_0}$. Thus, at initialization we can consider the inverse depth value $\tilde{\varrho}_0 = \varrho_0 + \boldsymbol{\xi}_{\varrho_0}$, where $\boldsymbol{\xi}_{\varrho_0}$ is a zero mean Gaussian noise with variance $\sigma^2_{\varrho_0}$.

Thanks to the inverse depth it is possible to describe a point that lies on the viewing ray represented by the unit direction vector divided by the uncertain inverse depth $\frac{\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}}{\tilde{\varrho}}$. If we consider a non unitary direction vector, i.e., $\frac{\tilde{\mathbf{r}}^{(\mathcal{C})}}{\tilde{w}}$, the denominator does not describe directly the distance, but it acts as a scale on the module of the direction vector. In this case, we prefer to adopt the symbol w instead of ϱ, where the true distance can be recovered as $d = \frac{\|\bar{\mathbf{r}}\|}{w}$. It follows that $\tilde{w}_0 = w_0 + \boldsymbol{\xi}_{w_0}$ where $\sigma^2_{w_0}$ is the variance of $\boldsymbol{\xi}_{w_0}$.

A common choice for the standard deviation $\sigma_{\boldsymbol{\xi}_{\varrho}}$ of $\boldsymbol{\xi}_{\varrho}$ is to include the infinite distance, represented when the inverse depth is 0, in the $\pm k\sigma$ confidence interval of the depth $1/\tilde{\varrho}_0$. This implies that $\varrho_0 - k\sigma_{\boldsymbol{\xi}_{\varrho}} = 0$, i.e., $\sigma_{\boldsymbol{\xi}_{\varrho}} = \frac{\varrho_0}{k}$, where k is usually take between 2 and 3. This choice limits the minimum representable distance to $d_{min} = \frac{1}{\varrho_0 - k\sigma_{\boldsymbol{\xi}_{\varrho}}}$. A similar reasoning can be replicated for the inverse scaling factor.

Concluding, landmark addition involves two different terms that come as input: the image point perceived $\boldsymbol{p}_{img}$ and the guess for the initial inverse depth ϱ_0 or scale w_0. The elements are accompanied by two sources of uncertainty: the first is the noise on the image point position ($\boldsymbol{\xi}_{\boldsymbol{p}_{img}}$) and the second representing the uncertainty on the inverse depth ($\boldsymbol{\xi}_{\varrho}$) or scale ($\boldsymbol{\xi}_{w}$). The complete noise vector $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_{\boldsymbol{p}_{img}} \\ \boldsymbol{\xi}_{\varrho} \end{bmatrix}$ assumes typically the covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\xi}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma^2_{\varrho 0} \end{bmatrix}$.

As a final remarks, it has to be noticed that the Jacobian of the $K(\cdots)^{-1}$ function can not be directly expressed, since this function involve an iterative solution. However, the *inverse function theorem* [94] states that $\left.\frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\overline{\mathbf{y}}} = \left[\left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\overline{\mathbf{x}}} \right]^{-1}$ where $\overline{\mathbf{y}} = f(\overline{\mathbf{x}})$, i.e., the Jacobian of the inverse function $f(\cdot)$ evaluated in $f(\overline{\mathbf{x}})$ is the inverse of the Jacobian matrix of $f(\cdot)$ evaluated in $\overline{\mathbf{x}}$, if $\left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\overline{\mathbf{x}}}$ is not singular. Thus, the evaluation of the Jacobians of $K(\cdots)^{-1}$ can be performed by the inversion of the Jacobians of $K(\cdots)$.

6.4.1.2 Landmark Measurement

The landmark measurement function aims at computing the predicted projection of a point in the image by properly composing the current robot position $\mathbf{\Gamma}_k$ and a landmark $\mathbf{y}_i$ to compute the point projection in the camera. The measurement is performed by a template matching between the stored patch that describes a landmark and the zone of the image around the predicted measure. The result of the template matching mechanism is affected by noise, that is considered a zero mean additive Gaussian noise $\boldsymbol{\delta}$ on the predicted projection. The covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\delta}}$ is commonly chosen as an identity 2×2 matrix, representing an independent uncertainty on the two coordinates of the template matching results with standard deviation of 1 pixel.

6.4.2 Inverse Scaling or Homogeneous Point - IS

The *Inverse Scaling* parameterization was presented in [62]. Although it was not the first parameterization to be introduced, it is presented as the first here because it has significantly different characteristic from all others parameterization.

6.4.2.1 Definition

A 3D point is represented in IS parameterization with an homogeneous 3D point, i.e., with a 4-elements vector

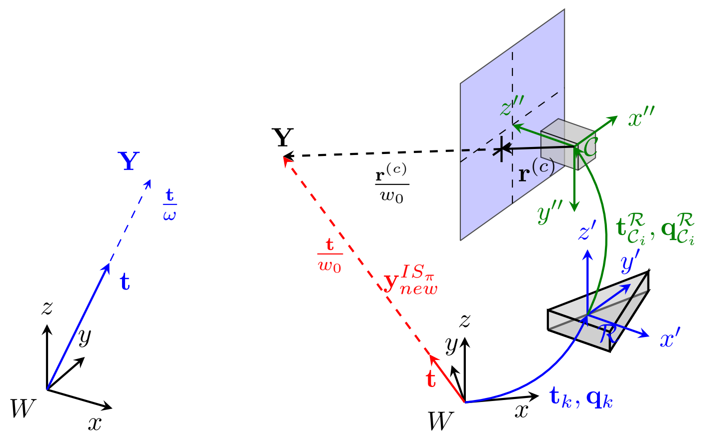
$$\mathbf{y}^{IS} = \begin{bmatrix} \mathbf{t} \\ w \end{bmatrix}, \tag{6.19}$$

where $\mathbf{t}$ is a vector and w is an inverse scale factor applied to it, as represented in Figure 6.1(a).

6.4.2.2 Euclidean Point

The transformation of an IS landmark, depicted in Figure 6.1(a), to a 3D point is simply given by the ratio between the $\mathbf{t}$ vector and the scaling factor w:

$$\mathbf{Y} = p^{IS}(\mathbf{y}^{IS}) = \frac{\mathbf{t}}{w}. \tag{6.20}$$



$\mathbf{Y}$
$\frac{\mathbf{t}}{\omega}$
$\mathbf{t}$
z
y
W
x
$\mathbf{Y}$
$\frac{\mathbf{r}^{(c)}}{w_0}$
$\mathbf{r}^{(c)}$
z''
x''
$\mathcal{C}$
y''
$\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}, \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}$
z'
y'
$\frac{\mathbf{t}}{w_0}$
$\mathbf{y}^{IS_\pi}_{new}$
$\mathcal{R}$
x'
z
y
$\mathbf{t}$
W
x
$\mathbf{t}_k, \mathbf{q}_k$

(a) Definition (b) Initialization

Figure 6.1: *IS parameterization. (a) definition and transformation to a 3D point (b) initialization with the* $g^{IS_\pi}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{p}_{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, w_0\}, \boldsymbol{\xi})$ *function.*

The evaluation of the Jacobian of $p(\cdot)$ with respect to the landmark allows to know the covariance ellipsoid of the 3D point, apart from the approximation introduced by linearizations, through $\frac{\partial p^{IS}(\mathbf{y})}{\partial \mathbf{y}} \mathbf{\Sigma}_{\mathbf{yy}} \frac{\partial p^{IS}(\mathbf{y})}{\partial \mathbf{y}}^T$, where the Jacobian is evaluated at the current estimation of the landmark and $\mathbf{\Sigma}_{\mathbf{yy}}$ is the block element on the diagonal of the covariance matrix corresponding to the landmark.

6.4.2.3 Initialization

To initialize a landmark, that is located on the viewing ray referenced in the camera reference frame $\mathbf{r}^{(\mathcal{C})}$ at an unknown distance, we have to compose the current robot pose in the world reference system and the transformation between the robot and the camera reference frames. Notice that in the formulas, in order to specify completely the initialization function, we are considering the viewing ray $\tilde{\mathbf{r}}^{(\mathcal{C})}$ affected by the noises $\boldsymbol{\xi}_{\boldsymbol{p}_{img}}$ and the initial depth $\tilde{w}_0$ with the zero mean additive noise $\boldsymbol{\xi}_w$. This results in the following homogeneous transformation, that can be followed with the help of Figure 6.1(b),

$$\begin{bmatrix} \mathbf{R}(\mathbf{q}_k) & \mathbf{t}_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}) & \mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^{(\mathcal{C})} \\ \tilde{w}_0 \end{bmatrix}, \tag{6.21}$$

that generates the following initialization function

$$\begin{aligned} \mathbf{y}^{IS_\pi}_{new} &= g^{IS_\pi}(\mathbf{\Gamma}_k, \mathbf{s} = \{\boldsymbol{p}_{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, w_0\}, \boldsymbol{\xi}) = \\ &= \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})\tilde{\mathbf{r}}^{(\mathcal{C})} + \tilde{w}_0\mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \tilde{w}_0\mathbf{t}_k \\ \tilde{w}_0 \end{bmatrix}. \end{aligned} \tag{6.22}$$

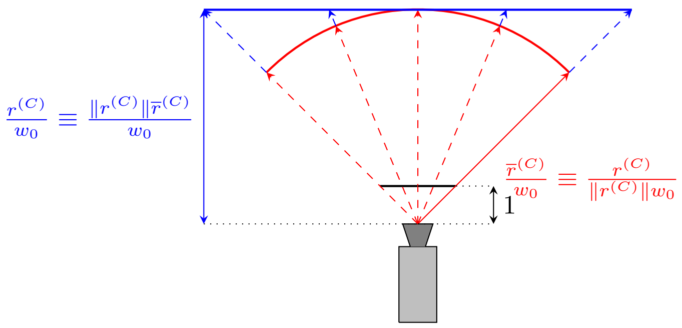

$\frac{r^{(C)}}{w_0} \equiv \frac{\|r^{(C)}\|\overline{r}^{(C)}}{w_0}$
$\frac{\overline{r}^{(C)}}{w_0} \equiv \frac{r^{(C)}}{\|r^{(C)}\|w_0}$
1

Figure 6.2: *Different initialization schema for the new landmarks: in red, landmarks are assumed initially distributed at equal distance from the camera center, in blue, landmarks are initialized on a plane parallel to the image plane.*

Notice that the $\mathbf{r}^{(\mathcal{C})}$ goes from the camera center to the normalized image plane, thus it has the z coordinate equal to 1. This implies that all the new landmark are initialized on a plane at a fixed distance $1/w_0$ from the normalized image plane.

A common choice in the initialization is to assume that points lies on a sphere centered on the camera center. This results in the two following possible initialization functions

$$
\begin{aligned}
\mathbf{y}_{new}^{IS_N} &= g^{IS_N}(\boldsymbol{\Gamma}_k, \mathbf{s} = \left\{\boldsymbol{p}_{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, w_0\right\}, \boldsymbol{\xi}) = \\
&= \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})} + \tilde{w}_0\mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \tilde{w}_0\mathbf{t}_k \\ \tilde{w}_0 \end{bmatrix},
\end{aligned} \tag{6.23}
$$

$$
\begin{aligned}
\mathbf{y}_{new}^{IS_S} &= g^{IS_N}(\boldsymbol{\Gamma}_k, \mathbf{s} = \left\{\boldsymbol{p}_{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, w_0\right\}, \boldsymbol{\xi}) = \\
&= \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})\tilde{\mathbf{r}}^{(\mathcal{C})} + \|\mathbf{r}^{(\mathcal{C})}\|\tilde{w}_0\mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \|\mathbf{r}^{(\mathcal{C})}\|\tilde{w}_0\mathbf{t}_k \\ \|\mathbf{r}^{(\mathcal{C})}\|\tilde{w}_0 \end{bmatrix}.
\end{aligned} \tag{6.24}
$$

The difference between the two initialization relies on the use respectively of the unitary viewing ray $\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}$ and the use of the viewing ray $\tilde{\mathbf{r}}^{(\mathcal{C})}$. The scaling factor is properly adapted, thanks to the equivalence $\frac{\bar{\mathbf{r}}^{(\mathcal{C})}}{w_0} \equiv \frac{\mathbf{r}^{(\mathcal{C})}}{\|\mathbf{r}^{(\mathcal{C})}\|w_0}$. Notice that it is possible to define a fourth initialization function $g^{IS_{\pi s}}$ that initialize point on the parallel plane using the unitary vector $\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}$ and the scaled inverse distance $\frac{w_0}{\|\mathbf{r}^{(\mathcal{C})}\|}$, since $\frac{\tilde{\mathbf{r}}^{(\mathcal{C})}}{w_0} \equiv \frac{\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}}{\frac{w_0}{\|\mathbf{r}^{(\mathcal{C})}\|}}$.

Summarizing, the two different approaches for the initialization produces an initial distribution of landmark on the surface of a sphere centered in the camera reference frame or on a plane parallel to the image plane (i.e., to the normalized image plane), as clarified by Figure 6.2.

Lets take a look to the Jacobians of the three initialization functions with respect to the state variables and the noise variables:

$$\mathbf{G}_{\mathbf{\Gamma}}^{IS_\pi} = \frac{\partial g^{IS_\pi}(\cdots)}{\partial(\mathbf{t}_k, \mathbf{q}_k)} = \begin{bmatrix} \tilde{w}_0 \mathbf{I} & \frac{\partial\left(\mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\mathbf{r}}^{(\mathcal{C})} + \tilde{w}_0 \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right)}{\partial \mathbf{q}_t^T} \\ 0 & 0 \end{bmatrix}, \tag{6.25}$$

$$\mathbf{G}_{\boldsymbol{\xi}}^{IS_\pi} = \frac{\partial g^{IS_\pi}(\cdots)}{\partial(\boldsymbol{\xi}_{\boldsymbol{p}_{img}}, \boldsymbol{\xi}_w)} = \begin{bmatrix} \frac{\partial \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\mathbf{r}}^{(\mathcal{C})}}{\partial \boldsymbol{p}_{img}^T} & \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \mathbf{t}_k \\ 0 & 1 \end{bmatrix}, \tag{6.26}$$

$$\mathbf{G}_{\mathbf{\Gamma}}^{IS_N} = \frac{\partial g^{IS_N}(\cdots)}{\partial(\mathbf{t}_t^T, \mathbf{q}_t)} = \begin{bmatrix} \tilde{w}_0 \mathbf{I} & \frac{\partial\left(\mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})} + \tilde{w}_0 \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right)}{\partial \mathbf{q}_t^T} \\ 0 & 0 \end{bmatrix}, \tag{6.27}$$

$$\mathbf{G}_{\boldsymbol{\xi}}^{IS_N} = \frac{\partial g^{IS_N}(\cdots)}{\partial(\boldsymbol{\xi}_{\boldsymbol{p}_{img}} \boldsymbol{\xi}_w)} = \begin{bmatrix} \frac{\partial \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\bar{\mathbf{r}}}^{(\mathcal{C})}}{\partial \boldsymbol{p}_{img}^T} & \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \mathbf{t}_k \\ 0 & 1 \end{bmatrix}, \tag{6.28}$$

and

$$\begin{aligned} \mathbf{G}_{\mathbf{\Gamma}}^{IS_S} &= \frac{\partial g^{IS_S}(\cdots)}{\partial(\mathbf{t}_t^T, \mathbf{q}_t)} = \begin{bmatrix} \tilde{w}_0 \|\mathbf{r}^{(\mathcal{C})}\| \mathbf{I} & \frac{\partial\left(\mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\mathbf{r}}^{(\mathcal{C})} + \tilde{w}_0 \|\mathbf{r}^{(\mathcal{C})}\| \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right)}{\partial \mathbf{q}_t^T} \\ 0 & 0 \end{bmatrix}, \\ \mathbf{G}_{\boldsymbol{\xi}}^{IS_S} &= \frac{\partial g^{IS_S}(\cdots)}{\partial(\boldsymbol{\xi}_{\boldsymbol{p}_{img}} \boldsymbol{\xi}_w)} = \begin{bmatrix} \frac{\partial \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})\tilde{\mathbf{r}}^{(\mathcal{C})}}{\partial \boldsymbol{p}_{img}^T} & \|\mathbf{r}^{(\mathcal{C})}\| \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \|\mathbf{r}^{(\mathcal{C})}\| \mathbf{t}_k \\ 0 & \|\mathbf{r}^{(\mathcal{C})}\| \end{bmatrix}. \end{aligned} \tag{6.29}$$

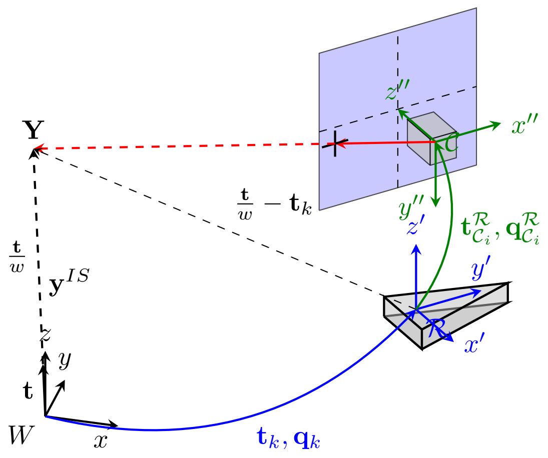
It should be noted that the Jacobians of the two initialization functions are perfectly equivalent, but this is true only because we use the module of the nominal viewing ray $\|\mathbf{r}^{(\mathcal{C})}\|$ to normalize the viewing ray or to scale the initial inverse depth factor. If it had not been done, spurious correlation between the initial depth uncertainty and the point direction would be introduced. This can be shown by calculating the bottom left element of the $\mathbf{G}_{\boldsymbol{\xi}}^{IS_S}$ matrix by using $\|\tilde{\mathbf{r}}^{(\mathcal{C})}\|$ as normalization factor instead of $\|\mathbf{r}^{(\mathcal{C})}\|$ in the function definition and noticing that it differs from zero.

It has to be pointed out that the original proposal of [62] use a different initialization function for this parameterization, where the undistorted coordinate of the image point are used together with the focal length to describe the direction vector. Here we prefer to present this formulation, using the normalized viewing ray or the point coordinate on the normalized image plane, to be more coherent with the parameterization that will be presented in the following.

6.4.2.4 Measurement function

The IS parameterization measurement equation, shown in Figure 6.3 that aims at the calculation of the predicted location of the projection of the landmark in an image, is:

$$\begin{aligned} &h^{IS}(\mathbf{\Gamma}_k, \mathbf{y}_k, \mathbf{v} = \left\{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\right\}, \boldsymbol{\delta}) = \\ &\quad = K_i \left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{R}(\mathbf{q}_k)^T \left(\frac{\mathbf{t}}{w} - \mathbf{t}_k \right) - \mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} \right) + \boldsymbol{\delta}. \end{aligned} \tag{6.30}$$



$\mathbf{Y}$
z''
x''
$\frac{\mathbf{t}}{w} - \mathbf{t}_k$
y''
z'
$\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}, \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}$
$\frac{\mathbf{t}}{w}$
$\mathbf{y}^{IS}$
y'
$\mathcal{R}$
x'
z
y
$\mathbf{t}$
W
x
$\mathbf{t}_k, \mathbf{q}_k$

Figure 6.3: *Measurement of a IS landmark from the current camera position.*

$\frac{\mathbf{t}}{w} - \mathbf{t}_k$ is a vector, expressed in the world reference system, that goes from the current robot pose to the landmark position. This vector is rotated by $\mathbf{R}(\mathbf{q}_k)^T$ to be expressed in the current robot reference system and successively by $\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T$ to be expressed in the camera reference frame. By subtracting $\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}$ from the resulting vector we obtain the vector that goes from the current camera position to the landmark, expressed in the camera reference frame. The application of the $K_i(\cdot)$ function to the vector computes the image point that is the predicted projection of the landmark in the image, that is the goal of the measurement function. Finally, the noise $\boldsymbol{\delta}$ is added to model the uncertainty on the matching process.

The measurement function can be rewritten, thanks to the properties of invariance of the K function, as

$$\begin{aligned} h^{IS}(\boldsymbol{\Gamma}_k, \mathbf{y}_k, \mathbf{v} = \left\{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\right\}, \boldsymbol{\delta}) = \\ = K_i\left(\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T\mathbf{R}(\mathbf{q}_k)^T(\mathbf{t} - w\mathbf{t}_k) - w\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}\right) + \boldsymbol{\delta}, \end{aligned} \tag{6.31}$$

to avoid division by zero when points are at the infinity ($w = 0$). Notice that a landmark can be measured in a different camera with respect to the one used in the initialization only by using the proper transformation between the camera and the robot $\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}$ and the proper K_i function. This constitute an easy way to predict and measure landmarks projection in a multi-camera rig system.

6.4.2.5 Transformation to a different reference frame

In order to use this parametrization in the CI SLAM framework, it is necessary to specify how to transform the landmarks in a different reference system through the function $lt(\boldsymbol{\Gamma}_k^{-1}, \mathbf{y})$, introduced in Section 5.6.2.1 through Equation 5.86. In this case, we can use the transformation of the homogeneous point represented by the landmark variable $\mathbf{y}$, resulting in

$$\begin{bmatrix} \mathbf{R}(\mathbf{q}_k)^T & -\mathbf{R}(\mathbf{q}_k)^T\mathbf{t}_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)^T\mathbf{t} - w\mathbf{R}(\mathbf{q}_k)^T\mathbf{t}_k \\ w \end{bmatrix}, \tag{6.32}$$

and the use of the Jacobians with respect to $\mathbf{\Gamma}_k$ and $\mathbf{y}$ allows to propagate properly the covariance.

6.4.3 Unified Inverse Depth - UID

The Unified Inverse Depth parameterization was proposed by Montiel et al. in [65] and thanks to its introduction it was possible to abandon the delayed initialization of the landmarks in the filter approach and to introduce a landmark despite its depth being completely unknown.

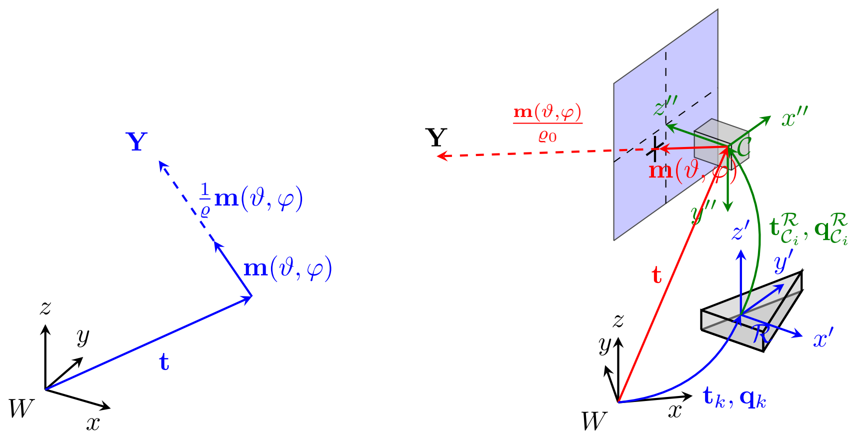
6.4.3.1 Definition

A 3D scene point $\mathbf{y}^{UID}$ can be defined through the 6 element vector

$$\mathbf{y}^{UID} = \begin{bmatrix} \mathbf{t} \\ \vartheta \\ \varphi \\ \varrho \end{bmatrix}, \tag{6.33}$$

where $\mathbf{t}$ is a 3-element vector representing the position of the camera center with respect to the world reference frame when the landmark was first perceived; ϑ and φ are two angles in radians representing the azimuth and the elevation of a direction vector applied to the $\mathbf{t}$ position; ϱ represents the inverse depth of the point on the direction vector, as shown in Figure 6.4(a).

Before continuing the explanation, we point out a particular characteristic of an UID landmark: the $\mathbf{t}$ element represents the position of the camera center with respect to the world reference frame when the landmark was firstly perceived. It acts as an *anchor point* (as noticed in [90]) on the landmark parameterization; thus, UID parameterization is said to be *anchored*, while the IS parameterization presented in the previous section is not anchored. If more than one landmark is initialized at the same time, i.e., it is measured in the same image for the first time, the anchor point element can be shared between all the newly added landmarks. Thus the anchor point is added to the filter only one time. Thanks to this shrewdness, for each new frame containing new landmarks, the state vector grows of $3 + 3n$ elements, being n the number of landmarks added in that frame. This concept was noticed in [43] and [74], but it assumes a greater importance in the CI SLAM framework, although it was not clearly explained in the reference papers. It can be shown that multiple copies of the same element in an EKF filter correspond to duplicate block of rows and columns in the covariance matrix, due to the fact that the copied elements are perfectly correlated, i.e., they are linearly dependent. This implies that if the shared version of the UID parameterization is not used, the CI SLAM framework can not be used, since the inversion of the covariance matrix relative to the shared landmarks between maps in the backpropagation step may fail due to singularity.



Y
$\frac{1}{\varrho}\mathbf{m}(\vartheta,\varphi)$
$\mathbf{m}(\vartheta,\varphi)$
z
y
$\mathbf{t}$
W
x
Y
$\frac{\mathbf{m}(\vartheta,\varphi)}{\varrho_0}$
z''
x''
$\mathcal{C}$
$\mathbf{m}(\vartheta,\varphi)$
y''
z'
$\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}, \mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}}$
y'
$\mathbf{t}$
$\mathcal{R}$
x'
z
y
$\mathbf{t}_k, \mathbf{q}_k$
x
W

(a) Definition (b) Initialization

Figure 6.4: *UID parameterization with anchor point. (a) definition and transformation to a 3D point (b) initialization with the* $g^{AP}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\})$ *function for the anchor point and* $g^{UID}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{p}_{img}, \mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}, \varrho_0\}, \boldsymbol{\xi})$ *function for the UID proper part.*

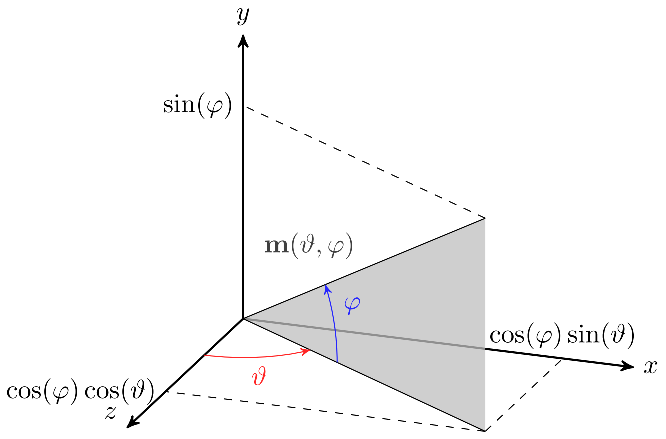
For the afore mentioned reason, here we split the definition of the UID parametrization in two parts: the definition of the anchor point (AP) and the definition of the proper UID part. The two parts can be added to the filter as two different landmarks: the anchor point landmark is added when at least one landmark has to be added at the current time step, then all the UID landmarks are added. Implementations have to keep trace of the correspondences between an anchor point and the UID landmarks anchored to it. The anchor point is simply represented by a 3-elements vector

$$\mathbf{y}^{AP} = \left[\mathbf{t}\right], \tag{6.34}$$

while the UID parameterization is redefined as the remaining 3 elements of the original proposal:

$$\mathbf{y}^{UID} = \begin{bmatrix} \vartheta \\ \varphi \\ \varrho \end{bmatrix}. \tag{6.35}$$

It has to be noticed that the introduction of the shared anchor point introduces some modifications in the standard algorithm presented for the EKF SLAM. The landmark addition function is duplicated into the anchor point addition and to the proper landmark addition. The landmark transformation function is duplicated too since both the anchor point and the landmark have to be referred to the new reference frame. The landmark measurement function has to take in input both the anchor point and the proper landmark, thus its Jacobian matrix is still very sparse, but with three non-zero blocks instead of two.



y
$\sin(\varphi)$
$\mathbf{m}(\vartheta,\varphi)$
φ
$\cos(\varphi)\sin(\vartheta)$
x
ϑ
$\cos(\varphi)\cos(\vartheta)$
z

Figure 6.5: *Azimuth and elevation angles definition. The azimuth angle ϑ is defined in the $z-x$ plane, while the elevation φ is the angle between the vector and its projection on the $z-x$ plane.*

6.4.3.2 Euclidean Point

The transformation of an UID landmark $\mathbf{y}^{UID} = \begin{bmatrix}\vartheta & \varphi & \varrho\end{bmatrix}^T$, referred to its specific anchor point $\mathbf{y}^{AP} = \begin{bmatrix}\mathbf{t}\end{bmatrix}$ into a 3D point is performed by

$$\mathbf{Y} \quad = \quad p^{UID}(\mathbf{y}^{AP}, \mathbf{y}^{UID}) \quad = \quad \mathbf{t} + (1/\varrho)\cdot\mathbf{m}(\vartheta,\varphi), \tag{6.36}$$

where

$$\mathbf{m}(\vartheta,\varphi) = [\cos(\varphi)\sin(\vartheta), -\sin(\varphi), \cos(\varphi)\cos(\vartheta)]^T, \tag{6.37}$$

computes the unit vector from the azimuth and elevation angles, as shown in Figure 6.5. This unit vector is scaled by the inverse distance and then applied to the anchor point $\mathbf{t}$

6.4.3.3 Initialization of the Anchor Point

The anchor point is the position of the camera center when the landmark was firstly perceived, thus, it is necessary to compose the robot current pose $\mathbf{\Gamma}_k$ with the camera center translation $\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}$, obtaining

$$\mathbf{y}^{AP}_{new} = g^{AP}(\mathbf{\Gamma}_k, \mathbf{s} = \left\{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\right\}) = \left[\mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \mathbf{t}_k\right], \tag{6.38}$$

Notice that this function simply compose the current robot pose with the known transformation between robot and camera. No external input are involved in the anchor point addition but only variables that are already in the state vector, thus it is necessary to compute only the Jacobian $\mathbf{G}^{AP}_{\mathbf{\Gamma}} = \frac{\partial g^{AP}(\cdots)}{\partial(\mathbf{t}_k,\mathbf{q}_k)} = \begin{bmatrix}\mathbf{I} & \frac{\partial\mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}}{\partial\mathbf{q}_k}\end{bmatrix}$.

6.4.3.4 Initialization of the UID landmark

The initialization is performed by:

$$\mathbf{y}^{UID}_{new} = g^{UID}(\mathbf{\Gamma}_t, \mathbf{s} = \left\{\mathbf{p}^{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, \varrho_0\right\}, \boldsymbol{\xi}) = \begin{bmatrix} \theta\left(\tilde{\mathbf{r}}^{(\mathcal{W})}\right) \\ \phi\left(\tilde{\mathbf{r}}^{(\mathcal{W})}\right) \\ \tilde{\varrho}_0 \end{bmatrix}, \tag{6.39}$$

where

$$\tilde{\mathbf{r}}^{(\mathcal{W})} = \mathbf{R}(\mathbf{q}_k)\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})\tilde{\mathbf{r}}^{(\mathcal{C})} \ , \tag{6.40}$$

is the viewing ray of $\mathbf{p}_{img}$ in a frame that is oriented as the world reference frame, thanks to the composition of the robot attitude w.r.t. the world reference frame and the orientation of the camera w.r.t the robot, while

$$\theta(\mathbf{r}) = \text{atan2}(\mathbf{r}_x, \mathbf{r}_z); \ \phi(\mathbf{r}) = \text{atan2}(-\mathbf{r}_y, \sqrt{\mathbf{r}_x^2 + \mathbf{r}_z^2}) \ , \tag{6.41}$$

are the azimuth and elevation angles that describe the direction of the viewing ray in world coordinates.

The Jacobians of the initialization Equation 6.39 with respect to the camera pose $\mathbf{\Gamma}_k = [\mathbf{t}_k, \mathbf{q}_k]$ and the noises $\boldsymbol{\xi}$ are

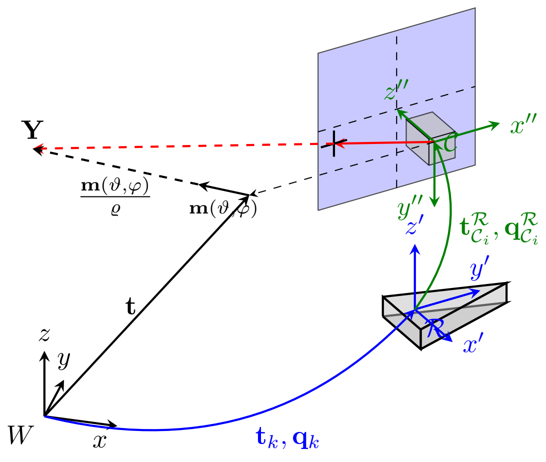
$$\mathbf{G}^{UID}_{\mathbf{\Gamma}} = \frac{\partial g^{UID}(\cdots)}{\partial(\mathbf{t}_k^T, \mathbf{q}_k)} = \begin{bmatrix} 0 & \frac{\partial\theta(\tilde{\mathbf{r}}^W)}{\partial\mathbf{q_t}^T} \\ 0 & \frac{\partial\phi(\tilde{\mathbf{r}}^W)}{\partial\mathbf{q_t}^T} \\ 0 & 0 \end{bmatrix}, \tag{6.42}$$

$$\mathbf{G}^{UID}_{\boldsymbol{\xi}} = \frac{\partial g^{UID}(\cdots)}{\partial(\boldsymbol{\xi}_{\mathbf{p}_{img}}\boldsymbol{\xi}_{\varrho})} = \begin{bmatrix} \frac{\partial\theta(\tilde{\mathbf{r}}^W)}{\partial\mathbf{p}^T_{img}} & 0 \\ \frac{\partial\phi(\tilde{\mathbf{r}}^W)}{\partial\mathbf{p}^T_{img}} & \\ 0 & 1 \end{bmatrix}. \tag{6.43}$$

The information about the sources of uncertainty related to the measurement process (i.e., the noise $\boldsymbol{\xi}_{p_{img}}$) and the uncertainty in the robot orientation are mixed together in the viewing ray, due to the fact that the $\theta(\cdot)$ and $\phi(\cdot)$ are represented in a world aligned frame, i.e., composed by the actual orientation of the robot with the direction of the viewing ray in the robot frame.

This initialization equation initializes points at a fixed distance from the camera center, i.e., on a sphere. To initialize points on a plane parallel to the image plane at distance $1/\rho_0$ we have to scale the initial inverse depth distance by the module of the viewing ray in world coordinates, obtaining

$$\mathbf{y}^{UID_\pi}_{new} = g^{UID}(\mathbf{\Gamma}_t, \mathbf{s} = \left\{\mathbf{p}^{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, \varrho_0\right\}, \boldsymbol{\xi}) = \begin{bmatrix} \theta\left(\tilde{\mathbf{r}}^{(\mathcal{W})}\right) \\ \phi\left(\tilde{\mathbf{r}}^{(\mathcal{W})}\right) \\ \|\mathbf{r}^{(\mathcal{C})}\|\tilde{\varrho}_0 \end{bmatrix}. \tag{6.44}$$



$\mathbf{Y}$
$\frac{\mathbf{m}(\vartheta,\varphi)}{\varrho}$
$\mathbf{m}(\vartheta,\varphi)$
$\mathbf{t}$
z''
x''
y''
$\mathcal{C}$
$\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}, \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}$
z'
y'
x'
$\mathcal{R}$
z
y
x
W
$\mathbf{t}_k, \mathbf{q}_k$

Figure 6.6: *Measurement of a UID landmark from the current camera position.*

6.4.3.5 Measurement function

The UID measurement equation uses the $\mathbf{y}^{UID}$ landmark and its corresponding anchor point $\mathbf{y}^{AP}$ to compute the predicted location of the projection of the landmark on an image:

$$\begin{aligned} h^{UID}(\mathbf{\Gamma}_k, \mathbf{y}^{AP}_k, \mathbf{y}^{UID}_k, \mathbf{v} = \left\{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\right\}, \boldsymbol{\delta}) = \\ = K_i \left(\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T \mathbf{R}(\mathbf{q}_k)^T \left(\mathbf{t} + \frac{\mathbf{m}(\vartheta,\varphi)}{\varrho} - \mathbf{t}_k - \mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} \right) \right) + \boldsymbol{\delta}. \end{aligned} \tag{6.45}$$

With reference to Figure 6.6, it can be noticed that the internal part of the equation is basically composed by vector composition operations: the term $\mathbf{t} + \frac{\mathbf{m}(\vartheta,\varphi)}{\varrho}$ sums two vector to reach the landmark location in world coordinate, then the current i-th camera position, resulting from the composition $\mathbf{t}_k + \mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}$ is subtracted from the landmark position to identify the viewing ray in world reference system. The viewing ray has to be transformed in the current camera reference frame and then the camera projection function is applied. The equation is rewritten and simplified to avoid division by zero when the landmark is at infinite distance as

$$\begin{aligned} h^{UID}(\mathbf{\Gamma}_k, \mathbf{y}^{AP}_k, \mathbf{y}^{UID}_k, \mathbf{v} = \left\{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\right\}, \boldsymbol{\delta}) = \\ = K_i \left(\mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T \mathbf{R}(\mathbf{q}_k)^T \left(\varrho \left[\mathbf{t} - \mathbf{t}_k\right] + \mathbf{m}(\vartheta,\varphi) \right) - \varrho \mathbf{R}(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i})^T \mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} \right) + \boldsymbol{\delta}. \end{aligned} \tag{6.46}$$

It has to be noticed that the Jacobian of this function with respect to the state variables is sparse and has three only non zero blocks, corresponding respectively to the derivatives with respect to the robot pose variable, the anchor point variable and the landmark parameterization.

6.4.3.6 Anchor Point Transformation In CI-SLAM

Alike for the IS parameterization, it is necessary to specify how to transform the landmarks in a different reference system through the function $lt(\mathbf{\Gamma}^{-1}_k, \mathbf{y})$, introduced in Section 5.6.2.1, Equation 5.86. For the UID parameterization we have to transform both the anchor point and the landmark.

The transformation of the anchor point represented by the anchor point variable $\mathbf{y}^{AP} = \mathbf{t}$ is

$$\left[\mathbf{R}(\mathbf{q}_k)^T (\mathbf{t} - \mathbf{t}_k)\right] \tag{6.47}$$

and through the use of the Jacobians we can propagate properly the covariance.

6.4.3.7 UID Transformation In CI-SLAM

The transformation of the UID part of the landmark requires to rotate the direction vector described by the ϑ and φ angles in the new reference system, while the inverse depth element remains unchanged. The new element results in

$$\begin{bmatrix} \theta\left(\mathbf{R}(\mathbf{q}_k)^T \mathbf{m}\left(\vartheta, \varphi\right)\right) \\ \phi\left(\mathbf{R}(\mathbf{q}_k)^T \mathbf{m}\left(\vartheta, \varphi\right)\right) \\ \varrho \end{bmatrix}. \tag{6.48}$$

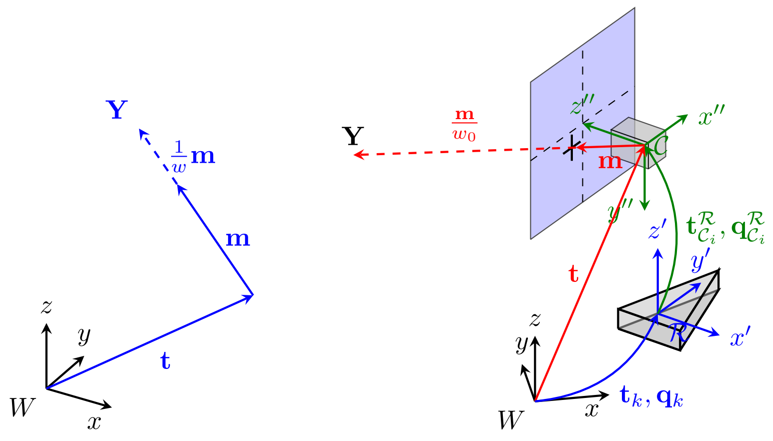
As usual, Jacobians have to be evaluated to compute the covariance matrix of the filter after the addition of the transformed version of the landmark.

6.4.3.8 UID to Euclidean 3D Point

A landmark represented with the UID parameterization can be converted into an Euclidean 3D point when a test on a *linearity index* is satisfied, i.e., when the uncertainty on the depth has been sufficiently reduced. This approach, which has been proposed in [14], aims to increase the computational efficiency of the UID parameterizations: in its original formulation, a UID landmark is represented with 6 elements, thus the conversion to a 3D point halves the occupied space in the filter state. However, we have presented the UID parameterizations in a different form by sharing the common anchor point between landmarks initialized at the same time. This implies that the conversion from UID to 3D points has no computational advantages, since a 3 elements UID landmark is substituted by a 3D point. When all the landmarks which share the same anchor points have been converted, the common anchor point can be removed from the filter, thus we obtain a reduction of the state vector dimension. It is clear that the more are the landmarks sharing the common anchor point, the less is the computational advantage which comes with the conversion. Similarly, the more are the landmarks sharing the common anchor point, the less is the probability that all of them will be converted. For these reasons, we will not treat anymore the conversion of landmarks to Euclidean 3D points, even if it can be easily applied to all the parameterizations presented in this work.

6.4.4 Anchored Homogeneous Point - AHP

The Anchored Homogeneous Point parameterization comes as an alteration of the UID parameterization. Although in its original formulation [90] it is presented as a monolithic parameterization with 7 elements per landmark, it shares with the UID parameterization the use of an anchor point. Thus, it changes with respect to the UID parameterization only in the landmark part, i.e., in the last 4 element.



(a) Definition

(b) Initialization

Figure 6.7: *AHP parameterization with anchor point. (a) definition and transformation to a 3D point (b) initialization with the* $g^{AP}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\})$ *function for the anchor point and* $g^{AHP_N}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{p}_{img}, \mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}, \varrho_0\}, \boldsymbol{\xi})$ *function for the AHP part.*

6.4.4.1 Definition

An AHP landmark alters the description of the direction vector by the use of a 3 element vector (not necessarily a unit vector) instead of the minimum representation given in UID by azimuth and elevation angles and it describe the depth by an inverse scale element. Thus, the AHP landmark, shown in Figure 6.7(a) is represented by

$$\mathbf{y}^{AHP} = \begin{bmatrix} \mathbf{m} \\ w \end{bmatrix}, \tag{6.49}$$

which corresponds to a direction vector $\mathbf{m}$ with a inverse depth scale factor w anchored to an anchor point $\mathbf{y}^{AP}$. It can be noticed that this corresponds to a 3D point represented in Homogeneous Coordinates; the anchoring to an anchor point leads to the name *Anchored Homogeneous Point*. Moreover, the AHP parameterization can be though as an evolution of the Inverse Scale parameterization, where an IS landmark is anchored to an anchor point.

6.4.4.2 Euclidean Point

The transformation of an AHP landmark $\mathbf{y}^{AHP} = \begin{bmatrix}\mathbf{m}^T & w\end{bmatrix}^T$, referred to its specific anchor point $\mathbf{y}^{AP} = \begin{bmatrix}\mathbf{t}\end{bmatrix}$ to a 3D point is performed by

$$\mathbf{Y} \quad = \quad p^{AHP}(\mathbf{y}^{AP}, \mathbf{y}^{AHP}) \quad = \quad \mathbf{t} + (1/w) \cdot \mathbf{m}. \tag{6.50}$$

It corresponds to the application of the direction vector, scaled by the inverse scale, to the anchor point.

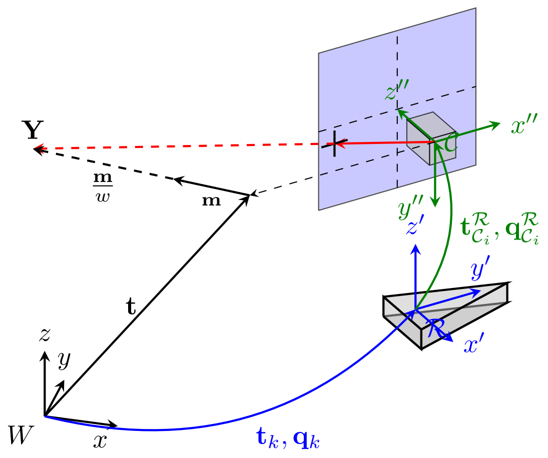


Figure 6.8: *Measurement of a AHP landmark from the current camera position.*

6.4.4.3 Initialization

An AHP point can be initialized in different ways. Let consider the more natural one

$$\mathbf{y}_{new}^{AHP_\pi} = g^{AHP_\pi}(\mathbf{\Gamma}_t, \mathbf{s} = \left\{\mathbf{p}^{img}, \mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}, w_0\right\}, \boldsymbol{\xi}) = \begin{bmatrix} \tilde{\mathbf{r}}^{(\mathcal{W})} \\ \tilde{w}_0 \end{bmatrix}, \tag{6.51}$$

which initializes the points on a plane parallel to the image plane at distance w_0. On the other hand, by using the unit vector in world coordinate $\tilde{\bar{\mathbf{r}}}^{(\mathcal{W})} = \frac{\tilde{\mathbf{r}}^{(\mathcal{W})}}{\mathbf{r}^{(\mathcal{C})}}$ as initialization for the $\mathbf{m}$ element, the initialization is performed on the sphere by the function $g^{AHP_N}(\cdots)$ (this function is shown in Figure 6.7(b). Similarly, it is possible to initialize the points on a plane by using the unitary direction vector and scaling properly the initial inverse depth ($g^{AHP_{\pi S}}(\cdots)$) or to initialize points on a sphere by using the non unitary direction vector and scaling the initial inverse depth ($g^{AHP_S}(\cdots)$).

6.4.4.4 Measurement Function

The measurement equation of an $\mathbf{y}^{AHP}$ landmark (Figure 6.8) applied to the anchor point $\mathbf{y}^{AP}$ is very similar to the UID landmark measurement equation. Here we propose only the final formula, where the inverse scale factor is at the numerator to avoid division by zeros:

$$\begin{aligned} h^{AHP}(\mathbf{\Gamma}_k, \mathbf{y}_k^{AP}, \mathbf{y}_k^{AHP}, \mathbf{v} = \left\{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\right\}, \boldsymbol{\delta}) = \\ = K_i \left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{R}(\mathbf{q}_k)^T \left(w\left[\mathbf{t} - \mathbf{t}_k\right] + \mathbf{m}\right) - w\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right) + \boldsymbol{\delta}. \end{aligned} \tag{6.52}$$

The only difference between AHP measurement equation and UID measurement equation lies in the direct presence of the direction vector $\mathbf{m}$ for AHP, while the direction vector has to be computed by the azimuth and elevation angles in UID.

6.4.4.5 AHP transformation In CI-SLAM

The transformation of the AHP landmark requires to rotate the direction vector $\mathbf{m}$ while the inverse depth element remains unchanged. The new element results in

$$\begin{bmatrix} \mathbf{R}(\mathbf{q}_k)^T\mathbf{m} \\ w \end{bmatrix}. \tag{6.53}$$

As usual, Jacobians has to be evaluated to calculate the covariance matrix after the addition of the transformed version of the landmark.

6.4.5 Framed Homogeneous Point - FHP

The Framed Homogeneous Point parameterization introduces an enhancement of the concept of anchored landmarks: it use a complete *anchor frame* instead of the anchor point. An anchor frame is defined as the camera position and orientation when a landmark was firstly perceived. Following the reasoning done for the anchor points, it is possible to add an unique anchor frame for more than one landmarks when they are initialized at the same time.

6.4.5.1 Anchor Frame Definition

The anchor frame is represented by the complete pose of the camera $\mathbf{\Gamma}$, i.e., it is composed by camera position $\mathbf{t}$ and camera orientation $\mathbf{q}$ w.r.t. the world reference frame. Thus, a landmark $\mathbf{y}^{AF}$ representing an anchor frame is composed by 7 elements:

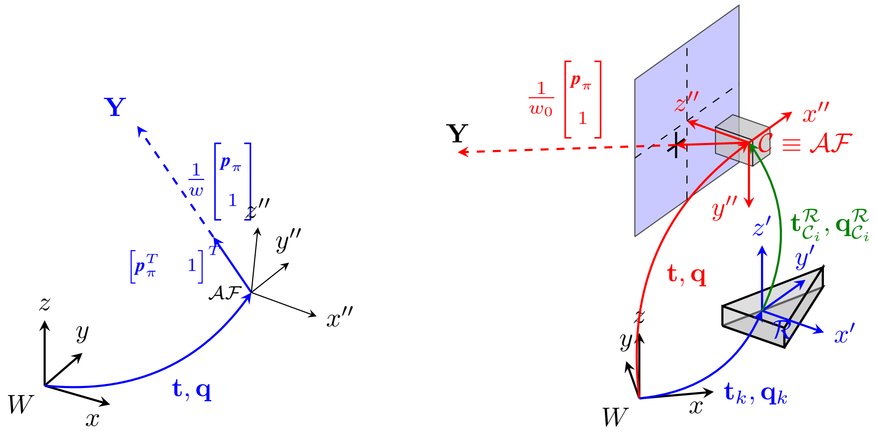
$$\mathbf{y}^{AF} = \mathbf{\Gamma} = \begin{bmatrix} \mathbf{t} \\ \mathbf{q} \end{bmatrix}. \tag{6.54}$$

It has to be noticed that during estimation the quaternions module may differ from 1, due to the update steps that can not guarantee to maintain the proper module. Obviously it is possible to normalize the quaternion with the formula of Equation 6.6 applied to each anchor frame, at a total cost of $O(n^2)$ being n the number of anchor frame in the filter. Assuming that an anchor frame has a scant movement during estimation, we can suppose that the modules of the quaternions representing the orientation of the anchor frames do not change a lot and that it is not necessary to normalize them continuously, saving computation time. The normalization is performed in the measurement equations of the framed parameterization.

6.4.5.2 Anchor Frame Initialization

The initialization of an anchor frame is done, given the current robot position $\mathbf{\Gamma}_k$ and the transformation between the robot and the camera $\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}$, by the composition of the two transformations

$$\mathbf{y}^{AF}_{new} = g^{AF}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{T}^{\mathcal{R}}_{\mathcal{C}_i}\}) = \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \mathbf{t}_k \\ \mathbf{q}_k \otimes \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i} \end{bmatrix}. \tag{6.55}$$



Y
$\frac{1}{w}\begin{bmatrix}p_\pi\\1\end{bmatrix}$
$\begin{bmatrix}p_\pi^T & 1\end{bmatrix}^T$
z''
y''
x''
$\mathcal{AF}$
z
y
x
W
$\mathbf{t},\mathbf{q}$
$\frac{1}{w_0}\begin{bmatrix}p_\pi\\1\end{bmatrix}$
$\mathcal{C}\equiv\mathcal{AF}$
$\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}},\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}}$
z'
y'
x'
$\mathcal{R}$
$\mathbf{t}_k,\mathbf{q}_k$

(a) Definition (b) Initialization

Figure 6.9: *FHP parameterization with anchor point. (a) definition and transformation to a 3D point (b) initialization with the* $g^{AF}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\})$ *function for the anchor frame and* $g^{FHP_\pi}(\mathbf{s} = \{\mathbf{p}^{img}, w_0\}, \boldsymbol{\xi})$ *function for the FHP proper part.*

As for the anchor point initialization function, no external inputs affected by uncertainty are used, thus the anchor frame is basically a copy of the current robot pose composed with the known transformation to obtain the current camera pose.

6.4.5.3 FHP Definition

The definition of the FHP parameterization with its anchor frame, is shown in Figure 6.9(a). Since the camera pose is completely specified by the anchor frame, the vector that specifies the landmark position can be used in the camera reference system and, as usual, an inverse scale factor is used to scale this vector. Moreover, the direction vector in camera reference system $\tilde{\mathbf{r}}^{(\mathcal{C})}$ has a 1 in its third component, since the application of $\tilde{\mathbf{r}}^{(\mathcal{C})}$ to the camera center reach the normalized image plane at distance 1 on the z axis. This implies that the third element can be excluded from the estimation and the 2D coordinate of the point on the normalized image plane (i.e., the first two element of $\tilde{\mathbf{r}}^{(\mathcal{C})}$), derived from $K^I\left(\boldsymbol{p}_{img}\right)^{-1}$) can be used to specify the direction vector. Thus, an FHP point is the 3 element vector:

$$\mathbf{y}^{FHP} = \begin{bmatrix}\boldsymbol{p}_\pi \\ w\end{bmatrix}. \tag{6.56}$$

6.4.5.4 Euclidean Point

An FHP landmark $\mathbf{y}^{FHP}$ with its corresponding anchor frame $\mathbf{y}^{AF}$ represents the 3D Euclidean point

$$\mathbf{Y} \quad = \quad p^{FHP}(\mathbf{y}^{AF}, \mathbf{y}^{FHP}) \quad = \quad \mathbf{t} + \frac{1}{w}\mathbf{R}\left(\frac{\mathbf{q}}{\|\mathbf{q}\|}\right)\begin{bmatrix}\boldsymbol{p}_\pi \\ 1.\end{bmatrix} \tag{6.57}$$

The initial camera position has to be summed to the scaled direction vector represented by the viewing ray in camera coordinate aligned with the initial camera orientation.

6.4.5.5 FHP Initialization

A FHP point can be initialized as

$$\mathbf{y}_{new}^{FHP_\pi} = g^{FHP_\pi}(\mathbf{s} = \left\{\mathbf{p}^{img}, w_0\right\}, \boldsymbol{\xi}) = \begin{bmatrix} K^I \left(\tilde{\boldsymbol{p}}_{img}\right)^{-1} \\ \tilde{w_0} \end{bmatrix}. \tag{6.58}$$

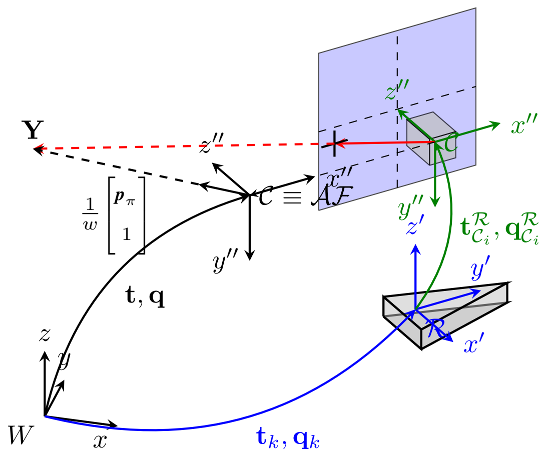
Using this initialization function, the points lie naturally on a plane parallel to the image plane at distance $1/w_0$, as shown in Figure 6.9(b). To obtain the spherical initialization it is necessary to scale the initial inverse depth $\|\mathbf{r}^{(\mathcal{C})}\|\tilde{w_0}$, obtaining the initialization function $g^{FHP_S}(\ldots)$.

It can be noticed that this initialization function, differently from what happens for the previous parameterization does not involve any variable from the state vector, but only externals inputs. This implies that the added landmark is initially uncorrelated by the rest of the filter. It becomes correlated thanks to successive measurements. This also imply that the FHP parameterization distinguish natively the uncertainty that comes from the camera pose, stored in the anchor frame landmark, and the uncertainty that comes from the perception of the landmark in the first image. We expect that this native separation can help in the estimation of the sub-pixel corrections on the viewing ray and in the camera pose, while parameterizations with anchor points can not distinguish the error in the initial camera orientation from the errors in the initial perception of the landmark in the image. It can be noticed that the parameterization results to be very similar to the one proposed in [43].

6.4.5.6 Measurement Function

The measurement equation of $\mathbf{y}^{FHP}$, shown in Figure 6.10, is very similar to the UID and AHP landmark measurement equations. We have to consider the initial camera orientation to rotate the direction vector, that is expressed in camera coordinates. Here we report the final formula, where the inverse scale factor is at the numerator to avoid division by zero:

$$h^{FHP}(\boldsymbol{\Gamma}_k, \mathbf{y}_k^{AF}, \mathbf{y}_k^{FHP}, \mathbf{v} = \left\{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\right\}, \boldsymbol{\delta}) =$$
$$= K_i\left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{R}\left(\mathbf{q}_k\right)^T\left(w\left[\mathbf{t}-\mathbf{t}_k\right] + \mathbf{R}\left(\frac{\mathbf{q}}{\|\mathbf{q}\|}\right)\begin{bmatrix}\boldsymbol{p}_\pi \\ 1\end{bmatrix}\right) - w\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right) + \boldsymbol{\delta}. \tag{6.59}$$



$\mathbf{Y}$
z''
x''
$\frac{1}{w}\begin{bmatrix}\boldsymbol{p}_\pi \\ 1\end{bmatrix}$
$\mathcal{C} \equiv \mathcal{AF}$
$\mathcal{C}$
y''
$\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i}, \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}$
z'
y'
x'
$\mathcal{R}$
$\mathbf{t}, \mathbf{q}$
z
y
W
x
$\mathbf{t}_k, \mathbf{q}_k$

Figure 6.10: *Measurement of a FHP landmark from the current camera position.*

6.4.5.7 Frame Transformation In CI-SLAM

Alike for the anchor point, the transformation of an anchor frame is needed when the CI-SLAM framework has to be used. The transformation of the anchor point represented by the landmark variable $\mathbf{y}^{AP} = \left[\mathbf{t}^T, \mathbf{q}^T\right]^T$ is

$$\begin{bmatrix} \mathbf{R}(\mathbf{q}_k)^T (\mathbf{t} - \mathbf{t}_k) \\ \mathbf{q}_k^* \otimes \frac{\mathbf{q}}{\|\mathbf{q}\|} \end{bmatrix}, \tag{6.60}$$

and the calculation of the Jacobians allows to propagate properly the covariance.

6.4.5.8 FHP Use In CI-SLAM

The proper part of the FHP parameterization does not require to be transformed in a different reference frame when used in the CI-SLAM framework with local submaps, but only to be shared (and copied during the map transition step) between maps. This comes from the fact that the direction vector in FHP is referred to the anchor frame and thus only the anchor frame has to be transformed.

6.4.6 Framed Inverse Scale - FIS

Analysing the FHP parametrization we can observe that the elements $\boldsymbol{p}_\pi$ represent, in the filter state, the initial point coordinates on the normalized image plane. Under the hypothesis of properly calibrated camera and a relatively high resolution, we can assume the initial point on the normalized plane to be properly initialized and we can assume it will not vary too much during updates. This implies that we can reduce the FHP parametrization only to the inverse depth element, obtaining the Framed Inverse Scale parameterization (FIS).

6.4.6.1 Definition

The FIS parameterization is

$$\mathbf{y}^{FIS} = \left[w\right], \tag{6.61}$$

and the initial point $\boldsymbol{p}_{img_0}$ in the image plane, that allows to recompute $\mathbf{p}_\pi$, is stored outside the filter state and it will not be updated. The FIS parameterization needs an anchor frame $\mathbf{y}^{AF}$ to be completely specified.

6.4.6.2 Euclidean point

A FIS landmark $\mathbf{y}^{FIS}$ with its corresponding anchor frame $\mathbf{y}^{AF}$ and the initial point $\boldsymbol{p}_{img_0}$ represents the 3D Euclidean point

$$\mathbf{Y} \;=\; p^{FIS}(\mathbf{y}^{AF}, \mathbf{y}^{FIS}, \mathbf{s} = \boldsymbol{p}_{img_0}, \boldsymbol{\xi}) \;=\; \mathbf{t} + (1/w)\mathbf{R}\left(\frac{\mathbf{q}}{\|\mathbf{q}\|}\right)\tilde{\mathbf{r}}_0^{(\mathcal{C})}, \tag{6.62}$$

where

$$\tilde{\mathbf{r}}_0^{(\mathcal{C})} \;=\; K^{-1}(\tilde{\boldsymbol{p}}_{img_0}), \tag{6.63}$$

$$\tilde{\boldsymbol{p}}_{img_0} \;=\; \boldsymbol{p}_{img_0} + \boldsymbol{\xi}. \tag{6.64}$$

The initial camera position $\mathbf{t}$ has to be summed to the scaled direction vector represented by the initial viewing ray $\tilde{\mathbf{r}}_0^{(\mathcal{C})}$ in camera coordinates rotated with the initial camera orientation $\mathbf{q}$. The initial viewing ray is computed through the use of the $K^{-1}(\cdot)$ function applied on the initial image point $\boldsymbol{p}_{img_0}$ perturbed by a bidimensional zero mean Gaussian noise $\boldsymbol{\xi}$. This noise is introduced to compensate the fact that the initial image point is not estimated using this parameterization, thus it remains uncertain as at the initialization. The covariance matrix of $\boldsymbol{\xi}$ can be assumed to be similar to the one used in the initialization function of the others parameterization, i.e., a diagonal matrix with 1px standard deviation.

6.4.6.3 Initialization

The initialization of the FIS parameterization needs only to specify the initial inverse depth, while the initial point $\boldsymbol{p}_{img_0}$ has to be stored outside the filter:

$$\mathbf{y}_{new}^{FIS_\pi} = g^{FIS_\pi}(\mathbf{s} = \{w_0\}, \boldsymbol{\xi}) = \left[\tilde{w}_0\right] \tag{6.65}$$

Notice that the $\boldsymbol{\xi}$ noise in this case is mono dimensional, i.e., it represent only the uncertainty on the inverse depth, since the uncertainty on the initial point is not needed at this stage. The spheric initialization can be easily obtained by scaling properly the initial depth with the module of the viewing ray, defining the function $g^{FIS_S}(\cdots)$.

6.4.6.4 Measurement Function

The measurement equation of a $\mathbf{y}^{FIS}$ landmark applied to the anchor frame $\mathbf{y}^{AF}$ is very similar to the FHP landmark measurement equation, but the initial point has to be used to compute the direction vector. Since the direction vector is not estimated with the FIS parameterization, we consider the noise $\boldsymbol{\delta}$ as a four element Gaussian noise: the first two elements are added to the resultant projection, while the last two are added to the initial point $\boldsymbol{p}_{img_0}$ on the image to represent the unmodeled uncertainty on the initial point in the image coordinates. The complete measurement equation results in

$$\begin{aligned} h^{FIS}(\mathbf{\Gamma}_k, \mathbf{y}_k^{AF}, \mathbf{y}_k^{FIS}, \mathbf{v} = \{\boldsymbol{p}_{img_0}, \mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\}, \boldsymbol{\delta}) = \\ = K_i \left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{R}\left(\frac{\mathbf{q}}{\|\mathbf{q}\|}\right)^T \left(w\left[\mathbf{t} - \mathbf{t}_k\right] + \mathbf{R}(\mathbf{q})\, \tilde{\mathbf{r}}_0^{(\mathcal{C})} \right) - w\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T \mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} \right) + \boldsymbol{\delta}_{12}, . \end{aligned} \tag{6.66}$$

where

$$\tilde{\mathbf{r}}_0^{(\mathcal{C})} = K^{-1}(\tilde{\boldsymbol{p}}_{img_0}), \tag{6.67}$$

$$\tilde{\boldsymbol{p}}_{img_0} = \boldsymbol{p}_{img_0} + \boldsymbol{\delta}_{34}, \tag{6.68}$$

is the viewing ray in camera reference system recomputed by the initial image point $\boldsymbol{p}_{img_0}$ with the addition of the last two elements of the Gaussian noise $\boldsymbol{\delta}$.

The 4×4 covariance matrix of δ is usually considered as a diagonal matrix. The first two element of the matrix are usually set to 1, to represent the uncertainty in the matching process, while the last two represent the uncertainty on the initial point image coordinates. A different choice is to consider exact the initialization by avoiding the insertion of the additional Gaussian noise $\boldsymbol{\delta}_{34}$. In fact, this additional noise can be interpreted as a *covariance inflation*, i.e., an artificial increase of the covariance of the measurement process which is used to compensate the unavoidable bias on the viewing ray, which is not continuously estimated. In the following, we will use the FIS$_0$ abbreviation to indicate that we are not considering the $\boldsymbol{\delta}_{34}$ noise in the measurement model.

As for the FHP parameterization, the FIS landmarks does not need to be transformed when a map transition of the CI-SLAM framework is performed, since they are expressed in the camera reference frame.

6.4.7 Remarks On The Parametrizations

Summarizing the characteristic of the parameterizations we can infer three categories: IS is the unique *unanchored* parameterizations; UID and AHP are *anchored to a point*; FHP and FIS are *anchored to a frame*.

The importance of the anchoring, either through point or frame, lies in the natural separation between the elements coming from the state vector and the elements that are proper of the perceived landmark. The unanchored parameterization mixes up in a single element both the camera pose (both position and orientation) and the measured direction of the landmark. The point anchored parameterizations separate the position in which the landmark has been measured for the first time (i.e., when it has been added to the filter), but mix the orientation of the camera with the measured direction of the landmark. The frame anchored parameterizations natively separate the camera pose from the perceived landmark direction vector.

In [90] it was argued that an anchored parameterizations performs better than the unanchored ones (i.e., the IS parameterization) due to the ability to account errors from the anchor point to the current robot position, instead of referring measures to the world frame. Following this reasoning, we can argue that frame anchored parameterizations maintain the same properties and they allow the introduction of a more precise separation, since the complete frame is maintained as an anchor. The more detailed measurement equations of framed parameterizations allow to distribute errors in a more effective way, i.e., to separate errors between the erroneous pose of the anchor frame, the erroneous perception of the landmark at first time and its uncertainty in the depth. This is the main reason that explains why inverse scaling parametrization performs worse than all others parametrization, as shown in [90] and [93]. We have to inform the interested reader that in the literature some different parameterizations have been presented, but they did not receive a significant attention by the scientific community, thus they are not considered in our work. Among them we cite [73], which uses the negative logarithm of the depth instead of the inverse depth representation, and [107], which develops a parallax parameterization for Bundle Adjustment algorithms that can be adopted in the EKF SLAM machinery too.

6.4.7.1 The special case of pure monocular SLAM

As a final remarks on the parameterization usage, we have to point out that when a pure monocular setup is used (i.e., when the odometric input is not used), the translation $\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}$, representing the camera position w.r.t. the robot reference frame has to be set to zero, i.e., we have to consider the robot reference system position coincident to the camera reference system position, while the rotation term $\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}}$ can be arbitrarily chosen. This limitation comes from the following consideration: the pure monocular SLAM system reconstruct the trajectory and the map up to an unknown scale factor s. Let consider a landmark addition equation, e.g., the anchor frame ones, reported in Equation 6.55:

$$\mathbf{y}_{new}^{AF} = g^{AF}(\mathbf{\Gamma}_k, \mathbf{s} = \{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\}) = \begin{bmatrix} \mathbf{t} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \mathbf{t}_k \\ \mathbf{q}_k \otimes \mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}} \end{bmatrix}. \tag{6.69}$$

It can be noticed that the translation part $(\mathbf{t})$ sums two elements that are not in the same scale: the transformation $\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}$ is metric, while the current robot pose $\mathbf{t}_k$ has a unknown scale. The right composition would have to be carried out as $\mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + s\mathbf{t}_k$, but this is not possible, since s is unknown. The erroneous composition does not affect the SLAM algorithm execution and the estimation of the robot trajectory, but prevents the proper estimation of the map, since the landmarks are referred to a erroneous anchor frame/point that can not be properly rescaled.

6.4.7.2 Impact Of Parameterizations On The State Size

Sharing the anchor part of the landmarks has great advantages in terms of dimension of the state vector, and, consequently, on the computational complexity; this is especially true for FIS parametrization. Let us initialize n landmarks in the same frame; we can notice that, for $n > 2$, FIS representation is more compact than UID, since $7+n < 3+3n$. Obviously, this is not valid for the FHP parametrization that has the same dimension of UID for the non shared part and a bigger shared part due to the use of an anchor frame instead of an anchor vector. On the other side, FHP state space is smaller than AHP for $n > 4$, since $7+3n < 3+4n$. In Table 6.1 we report how the vector state size increases at the addition of n landmarks. For $n > 2$ the FIS parameterization is the more economic one. We have to point out that this theoretical calculus may not be reflected to a real advantage in practical

Param.	Cost	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
IS	$4n$	**4**	**8**	12	16	20
UID	$3 + 3n$	6	9	12	15	18
AHP	$3 + 4n$	7	11	15	19	23
FHP	$7 + 3n$	10	13	16	19	22
FIS	$7 + n$	8	9	**10**	**11**	**12**

Table 6.1: *Comparison of the state dimension increase due to the addition of* n *landmark at the same time, i.e., sharing the same anchor point/frame. In bold are highlighted the less expensive parameterizations, i.e., IS for* $n \leq 2$ *and FIS for* $n > 2$.

applications. In real situations it may happen that some landmarks have to be deleted after the initialization because they can not be properly matched in the next images and this can reduce the number of landmarks that shares the same anchor point/frame. Moreover, we might not be able to add n points at the same time, although with $n = 2$ this is quite unlikely.

As a final remark, we notice how, by storing the full pose of the camera when the landmarks was perceived the first time, the FIS and FHP parametrizations implement some sort of key-framed representation embedded in the EKF machinery: every time a landmark gets updated by the Kalman filter, past camera poses get refined as well. This results in the ability of storing, inside the vector state a “smoothed trajectory”, constituted by the camera poses at time steps when at least a landmark has been added to the filter. Although the EKF-SLAM approach remains an *on-line* approach, i.e., it estimates the current robot pose and the static map, the particular representation of the map obtained by FHP and FIS contains a smoothed estimation of the previous robot poses, being somehow similar to the FrameSLAM [48] or GraphSLAM [103] approach.

6.4.7.3 Issues With The CI-SLAM Framework

The representation of the landmarks divided in shared part and proper part, i.e., the addition of an unique anchor point/frame for all the landmarks added at the same time, contributes to limit the growing rate of the state vector, saving computational time. Beside being an advantage from the computational point of view, it allows the use of the Conditional Independent Submapping [78] [77]). Indeed adding multiple copies of the anchor point/frame of the landmarks initialized at the same time would introduce fully correlated variables, resulting in a singular covariance matrix. For the back-propagation step of the CI-SLAM framework we need to invert the covariance matrix of the shared elements between maps and this would not be possible if more than one landmark added at the same time step is present in the shared elements. We can assert that the use of the shared representation of the landmarks is not only a shrewdness for efficiency, but a need for the usage of anchored parameterizations in the CI-SLAM framework.

The CI-SLAM framework suffers a second big issue that needs special attention. When a framed parameterization is used, the covariance matrix of the EKF tends to be close to singularity due to the presence of the quaternions. The latter are overparametrized, i.e., they uses four elements to represent the 3 dimensional space of rotations in 3D space. This implies that a relation, not necessarily linear, between the quaternions elements exists. In particular, the analysis of the eigenvalues and eigenvectors of a covariance matrix $\mathbf{\Sigma}$ that contains quaternions shows null eigenvalues $\lambda_i = 0$. Each eigenvector $\mathbf{x}_i$ corresponding to a null eigenvalues $\lambda_i = 0$ allows to write a linear equation $\mathbf{\Sigma x} = \lambda x = 0$, that confirm the singularity of the covariance matrix. It can be shown that each one of the quaternions in the filter is involved in a linear relation between its covariances elements. The usage of the CI-SLAM framework needs to avoid the singularity of the covariance matrix of the shared elements, but this is impossible when a framed parameterization is used, since it contains quaternions.

Here a partial solution to the problem that revealed to be insufficient for the complete usage of the CI-SLAM framework functionalities is presented. Considering a normalized quaternion $\mathbf{q} = [\mathbf{q}_w, \mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z]^T$, when it is sufficiently close to the identity, i.e., $[1, 0, 0, 0]^T$, we can represent it by only the axis part $[\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z]^T$ and reconstruct the fourth element as $\mathbf{q}_w = \sqrt{1 - \left(\mathbf{q}_x^2 + \mathbf{q}_y^2 + \mathbf{q}_z^2\right)}$. It can be noticed that the axis part representation of the small rotations is similar to the approximation for small rotations proposed in Equation 2.57 with the rotation vector. The CI-SLAM uses local maps, thus the quaternions in the filter, when framed parameterizations are used, are reasonably close to the identity. To use the framed parameterizations in the CI-SLAM framework, before a back-propagation step, we propose to normalize all the quaternions in the filter state, take the mean vectors and the covariance matrix $\mathbf{\Sigma}$ in the proposed reduced form, marginalizing out all the first term of the quaternions. Once the back-propagation has been performed the missing element of the quaternion can be reconstructed and the Jacobians of the reconstruction function allow to reconstruct the covariance matrix properly.

The proposed approach has been demonstrated effective when the simple backpropagation step is used, but it has shown its limit when the loop closure procedure of the CI-SLAM framework is used. Recalling Section 5.6.1.6, loop closure management needs to bring landmarks along all the maps involved in the loop, from the map i, where the features originally resides, to map j where the loop closure has take place. This implies that the rotation of the loop closing landmarks expressed in the reference frame of the intermediate maps may be very far from the identity, invalidating the hypothesis stated before. This consideration is the motivation that brings to the introduction of the *Hybrid EKF-SLAM* presented in the next chapter.

As a final remarks on the CI-SLAM framework, we point out that the map transition step can not be performed if some landmarks have been added at the last time step when anchored parameterizations are used. This comes from the fact that the anchor point (or frame) of the new landmarks is an exact copy of the robot position (or pose). When the anchor point/frame is composed with the inverse of the robot pose to obtain the anchor point/frame in the reference frame of the new map, its variance become zero, causing a singularity in the covariance matrix of the shared elements and preventing the possibility of the inversion. Two equivalent solution are possible: we can postpone the map transition until we observe that no landmark has been added at the current step or we can inhibit the landmark addition step when we need to perform the map transition.

CHAPTER 7

Indirect EKF SLAM

In this chapter we introduce a different way of thinking the EKF estimation mechanism, i.e., the so called *Indirect EKF* or *error state EKF*, and how this can be adapted to the EKF SLAM problem, introducing a formulation that we name *Hybrid EKF-SLAM*. The *Hybrid EKF-SLAM* formulation allows to solve the problems related to the usage of framed parameterizations in CI-SLAM framework, since rotations can be expressed in a minimal form close to the identity. In the remaining of this chapter the Indirect EKF formulation is presented, then the *Hybrid EKF-SLAM* formulation is derived and the motion models and the framed parameterizations defined in the previous chapter are adapted to the new formulation.

7.1 The Indirect or Error State EKF

In the conclusion of the previous chapter we have shown the problems related to the usage of the quaternions in the Conditional Independent EKF SLAM framework. Quaternions cause singular covariance matrix, since rotations are represented with an over-parametrized formulation. This implies that the back propagation step of the CI SLAM framework can not be applied directly, since covariance matrices have to be inverted. In Section 6.4.7.3 we proposed a workaround to this, that use the axis part of the quaternion only, assuming that represented rotations are small, i.e., close to the identity. Unfortunately, this hypothesis is infringed when loop closure are performed. This last issue can be solved by using an Indirect (or Error State) EKF instead of a standard Extended Kalman filter.

The *Indirect EKF* (also known as *error state EKF*) differs mainly from the standard EKF in how the system is modeled. In the standard formulation the state vector describes directly the state of the system (e.g. the pose of the robot, its velocity, etc.), thus the state transition function describes how the state evolves in time. In the indirect approach the state vector describes the *errors* with respect to nominal values. Errors and nominal values evolve in parallel respectively with error-state transition function and nominal-state transition function. Measurement equations involves both the nominal and the error state, updating the estimation of the errors. Errors can be reset and integrated in the nominal values, ensuring that the error state estimation remains always close to zero.

Lets introduce the mathematical notation that will be used hereafter. The nominal values of a variable are referred with the variable name $\mathbf{x}$, while the error part, i.e., the differences between the nominal values and the true values are indicated with $\delta\mathbf{x}$ and it will be indicated hereafter as the *delta* part. The state vector of the EKF contains only the error (delta) variables, since the nominal values are assumed to have no uncertainty. The *true value*, i.e., the full estimate state (the *total state*) of the system, $\tilde{\mathbf{x}}$ is then given by the composition of the nominal values and the error part

$$\tilde{\mathbf{x}} = \mathbf{x} \boxplus \delta\mathbf{x}, \tag{7.1}$$

where $\boxplus$ indicates a composition function that may differs from the algebraic sum. Notice that, thanks to this general formulation, it is not necessary for $\mathbf{x}$ and $\delta\mathbf{x}$ to have the same dimension.

All the operations of the EKF have to be specified both for the nominal values and for the error state. Consider for instance the motion model: it is necessary to specify the function $f_{\mathbf{D}}(\cdots)$ for the prediction of the nominal values and the function $\delta f_{\mathbf{D}}(\cdots)$ for the error part. Notice that the Jacobian computation and evaluation need to be done only on the latter function, since only it involves the state variables $\delta\mathbf{x}$ while nominal values are not involved in the estimation process. With reference to the general motion model of Equation 5.8, the complete definition of the motion model functions for the Indirect EKF-SLAM can be expressed as

$$\begin{bmatrix}\boldsymbol{\Gamma}_k \\ \boldsymbol{\Lambda}_k\end{bmatrix} = f_{\mathbf{D}}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}, \mathbf{u}_k), \tag{7.2}$$

$$\begin{bmatrix}\delta\boldsymbol{\Gamma}_k \\ \delta\boldsymbol{\Lambda}_k\end{bmatrix} = \delta f_{\mathbf{D}}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}, \delta\boldsymbol{\Gamma}_{k-1}, \delta\boldsymbol{\Lambda}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{7.3}$$

being Equation 7.2 the motion model for the nominal part and Equation 7.3 for the delta part. Similarly, landmark addition has to be defined both for the nominal values and for the error state. The measurement equation remains unique, but it involves in the parameters both the nominal values and the delta values.

There are two possible approaches to the development of an indirect EKF: the *feed-forward* indirect EKF and the *feed-back* indirect EKF. In the feed-forward approach the nominal values evolve independently and the delta values represent, at each time step, the estimation of the differences between the true state and the nominal values. In the feed-back approach the nominal values are corrected with the error estimation and the delta values are reset to zero. The latter is considered the best approach to the indirect filtering, since the values in the filter state are always maintained close to the zero, while in the feedforward approach they can grow unbounded.

Let explain how the delta values are reset and integrated in the nominal values when the feed-back approach is considered. First of all, when the composition function of the delta and nominal values (Equation 7.1) is simply the algebraic sum, the nominal values can be updated with the delta values, the delta values set to zero and the covariance matrix does not change, since a mean shift does not affect the covariance of a Gaussian variable. In general, composition is a non linear function and a general procedure to integrate the delta values in the nominal values, reset the delta and properly update the covariance matrix has to be developed. First of all, the new nominal values has to be computed by the composition of the current nominal values and the mean of the delta values

$$\mathbf{x}^{+} \quad = \quad \mathbf{x} \boxplus \overline{\delta\mathbf{x}}, \tag{7.4}$$

where the $\overline{\delta\mathbf{x}}$ are the current mean values of the delta values in the state vector. To reset the delta values and update properly the covariance matrix we can consider the following equation, derived from Equation 7.1 by considering two equivalent composition of different nominal and delta values:

$$\tilde{\mathbf{x}} \quad = \quad \mathbf{x}^{+} \boxplus \delta\mathbf{x}^{+} \quad = \quad \mathbf{x} \boxplus \delta\mathbf{x}. \tag{7.5}$$

In general, the same true value $\tilde{\mathbf{x}}$ can be expressed by different compositions of nominal and delta values, thus if we change the nominal values, we have to change the delta values according to them. Substituting the Equation 7.4 into Equation 7.5, i.e., considering the nominal value with the integrated mean of the delta values, we obtain

$$\left(\mathbf{x} \boxplus \overline{\delta\mathbf{x}}\right) \boxplus \delta\mathbf{x}^{+} \quad = \quad \mathbf{x} \boxplus \delta\mathbf{x}, \tag{7.6}$$

and, naming $\boxminus$ the inverse of $\boxplus$, by isolating the $\delta\mathbf{x}^{+}$ terms we obtain

$$\delta\mathbf{x}^{+} \quad = \quad \left(\mathbf{x} \boxplus \overline{\delta\mathbf{x}}\right) \boxminus \left(\mathbf{x} \boxplus \delta\mathbf{x}\right). \tag{7.7}$$

The last equation allows to compute properly the Jacobian

$$\left.\frac{\partial \delta\mathbf{x}^{+}}{\partial \delta\mathbf{x}}\right|_{\delta\mathbf{x}=\overline{\delta\mathbf{x}}}, \tag{7.8}$$

i.e., the Jacobian of the delta transformation function with respect to the current error state variable $\delta\mathbf{x}$ and to propagate covariance properly. Notice that the variable $\delta\mathbf{x}$ is, in general, just a part of a big state vector, thus the covariance propagation modifies only the vertical and horizontal bands of the covariance matrix corresponding to the variable $\delta\mathbf{x}$. This implies that the delta reset of each variable has $O(n)$ complexity, being n the number of variables in the filter state.

7.1.1 Hybrid EKF-SLAM

The use of an indirect EKF in a SLAM application, taking into account the CI SLAM framework too, needs some adjustment with respect to a standard indirect EKF system. Obviously, all the considerations about the sparsity of the Jacobians of the principal functions of the EKF are still valid, i.e., the prediction and the landmark addition steps can be performed in linear time with the number of landmarks, the Jacobians of the measurement equations are sparse and so on, but the delta reset procedure of the feedback approach add computational cost to the system.

In the choice between the feedback and the feed-forward approach, we tend to the first one, since it ensure to maintain all the variables in the vector state close to the zero, including rotations that can be represented with a minimal representation. A feed-forward representation does not offer the same guarantees, since the state vector can growth indefinitely. On the contrary, a straightforward application of the feedback indirect EKF to the SLAM problem may results in an excessively high computational cost. Reset the deltas of all the variables in the state vector has an additional cost of $O(n^2)$, being n the number of landmarks in the state. Although the overhead of this operation has the same complexity of the EKF SLAM algorithm, the computation time required by this operation may results excessively high for a real time system implementation. Since landmarks are static, we can suppose that the errors on their estimate remain sufficiently bounded during the estimation process. Thus, a *hybrid* approach may be used: the feed-back formulation is used for the dynamic part of the state vector, while landmarks are maintained in a feed-forward formulation, i.e., the delta of the landmarks are never reset. Consequently, the delta are reset only for the robot pose $\mathbf{\Gamma}_k$ and the motion parameters $\mathbf{\Lambda}_k$, while landmarks variables are never reset. We name this approach *Hybrid Indirect EKF-SLAM*. It can be noticed that the application of the delta reset procedure to the subset of the variables representing the dynamic of the robot bounds the complexity of this operation to $O(n)$.

There is a second motivation under the use of an hybrid approach between the feedback and feedforward formulation. If we suppose to use a pure feedback approach, i.e., to reset the delta of all the variables in the EKF state vector, including the landmarks, we can easily conclude that the CI-SLAM framework can not be applied directly. Considering the back-propagation Equation 5.107, it can be noticed that the differences between the mean values of the estimation of the shared variables in the two maps is performed on the delta values in the filter state. This implies that, if the deltas of the landmarks were reset, the difference become always 0 and the back propagation can not update the estimation of the shared landmark properly. Moreover, the covariances are linearized around a different point if the delta are reset, thus the covariance update in the back propagation will not be performed properly too. The presented hybrid approach fixes the initial nominal values of the landmarks and consequently the delta values are directly comparable between them, allowing the use of the CI SLAM procedures unchanged. Notice that the reset of the deltas on the motion parameters does not introduce any issue in the back propagation since the values in the new map are copied twice and the shared one remains static to take trace of the initial speed of the map and its delta are never reset.

In the remaining of this chapter, all the proposed function for the motion models and for the parameterizations are redrawn for the proposed *Hybrid Indirect EKF* formulation.

7.2 Motion Models

To define the motion models in the indirect form, we have to specify the two functions of Equations 7.2 and 7.3, which describes respectively the nominal motion model and the error (delta) motion model.

The true robot pose $\tilde{\mathbf{\Gamma}}$ is represented by a translation vector $\tilde{\mathbf{t}}$ and a quaternion $\tilde{\mathbf{q}}$. The translation nominal values and delta values are respectively represented by the vectors $\mathbf{t}$ and $\delta\mathbf{t}$. The true value of the translation is defined by the sum of the two vector, thus $\tilde{\mathbf{t}} = \mathbf{t} + \delta\mathbf{t}$. The orientation nominal values are represented by a quaternion $\mathbf{q}$ and the delta values are represented by the rotation vector $\delta\boldsymbol{\theta}$. Notice that in this case the delta values uses only 3 elements instead of the 4 used by the quaternion. The true value of the orientation is obtained through the small rotations approximation (see Section 2.3.5) as

$$\tilde{\mathbf{q}} = \mathbf{q} \otimes \delta\mathbf{q}, \tag{7.9}$$

being $\delta\mathbf{q} = \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}$.

The motion parameters $\tilde{\boldsymbol{\Lambda}}$, if used in the problem specification, are composed by the tangential and rotational velocity vectors $\tilde{\mathbf{v}}$ and $\tilde{\boldsymbol{\omega}}$. Composition of nominal and delta values are simply given as

$$\tilde{\mathbf{v}} = \mathbf{v} + \delta\mathbf{v}, \tag{7.10}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \delta\boldsymbol{\omega}. \tag{7.11}$$

The delta reset of variables which composition is done by sum is very simple. Let us consider the translation vector $\delta\mathbf{t}$; new nominal values can be computed as $\mathbf{t}^{+} = \mathbf{t} + \overline{\delta\mathbf{t}}$. The delta reset Equation 7.7 results in $\delta\mathbf{t}^{+} = \mathbf{t} + \delta\mathbf{t} - \mathbf{t} - \overline{\delta\mathbf{t}} = \delta\mathbf{t} - \overline{\delta\mathbf{t}}$ and it is easy to verify that the evaluation at $\delta\mathbf{t} = \overline{\delta\mathbf{t}}$ guarantees to reset the deltas. The Jacobian of the reset with respect to $\delta\mathbf{t}$ is the identity, thus the covariance is not affected by the delta reset procedure. Similarly this happens for $\mathbf{v}$ and $\boldsymbol{\omega}$ variables.

7.2.0.1 Quaternion delta reset function

The case of the quaternion is a little bit more complex. First of all, the new nominal value can be computed as $\mathbf{q}^{+} = \mathbf{q} \otimes \overline{\delta\mathbf{q}} = \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{\overline{\delta\boldsymbol{\theta}}}{2} \end{bmatrix}$. Let us write the Equation 7.5 as

$$\mathbf{q}^{+} \otimes \delta\mathbf{q}^{+} = \mathbf{q} \otimes \delta\mathbf{q}, \tag{7.12}$$

and, moving $\mathbf{q}^{+}$ to the right, we obtain, after some manipulation, the delta reset function of Equation 7.7

$$\begin{aligned} \delta\mathbf{q}^{+} &= \mathbf{q}^{+*} \otimes \mathbf{q} \otimes \delta\mathbf{q} && (7.13) \\ &= \left(\mathbf{q} \otimes \overline{\delta\mathbf{q}}\right)^{*} \otimes \mathbf{q} \otimes \delta\mathbf{q} && (7.14) \\ &= \overline{\delta\mathbf{q}}^{*} \otimes \mathbf{q}^{*} \otimes \mathbf{q} \otimes \delta\mathbf{q} && (7.15) \\ &= \overline{\delta\mathbf{q}}^{*} \otimes \delta\mathbf{q}. && (7.16) \end{aligned}$$

Then it is possible to substitute the delta quaternion with the rotation vector, obtaining

$$\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}^{+}}{2} \end{bmatrix} = \eta \begin{bmatrix} 1 \\ -\frac{\overline{\delta\boldsymbol{\theta}}}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}, \tag{7.17}$$

where η is a normalization factor that guarantees that the scalar value of the resulting quaternion is equal to 1. This equation has to be manipulated as follow to obtain an explicit expression for η and $\delta\theta^{+}$.

$$\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}^{+}}{2} \end{bmatrix} = \eta \mathbf{Q}\left(\begin{bmatrix} 1 \\ -\frac{\overline{\delta\boldsymbol{\theta}}}{2} \end{bmatrix}\right)\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} \tag{7.18}$$

$$= \eta\left(\mathbf{I} + \begin{bmatrix} 0 & \frac{\overline{\delta\boldsymbol{\theta}}}{2}^{T} \\ -\frac{\overline{\delta\boldsymbol{\theta}}}{2} & \left[\frac{\overline{\delta\boldsymbol{\theta}}}{2}\right]_{\times} \end{bmatrix}\right)\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} \tag{7.19}$$

$$= \eta\left(\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} + \begin{bmatrix} \frac{\overline{\delta\boldsymbol{\theta}}}{2}^{T}\frac{\delta\boldsymbol{\theta}}{2} \\ -\frac{\overline{\delta\boldsymbol{\theta}}}{2} + \left[\frac{\overline{\delta\boldsymbol{\theta}}}{2}\right]_{\times}\frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}\right), \tag{7.20}$$

where the quaternion multiplication is expressed with the 4×4 matrix $\mathbf{Q}(\cdot)$. From the first element of the vector it is possible to compute the normalization factor

$$\eta = \frac{1}{1+\frac{\overline{\delta\boldsymbol{\theta}}}{2}^{T}\frac{\delta\boldsymbol{\theta}}{2}}, \tag{7.21}$$

and then the last three elements can be computed as

$$\delta\theta^{+} = \eta\left(\delta\boldsymbol{\theta} - \overline{\delta\boldsymbol{\theta}} + \left[\frac{\overline{\delta\boldsymbol{\theta}}}{2}\right]_{\times}\delta\boldsymbol{\theta}\right). \tag{7.22}$$

The function resets the deltas in the state when evaluated in $\delta\boldsymbol{\theta} = \overline{\delta\boldsymbol{\theta}}$ and the computation of the Jacobians w.r.t $\delta\boldsymbol{\theta}$ allows to propagate properly the covariance, i.e., to linearize it around the reset deltas.

7.2.1 Motion Model With Odometric Informations

Lets specify the motion model in the indirect form when the odometric information is available. The prediction function for the nominal values

$$\boldsymbol{\Gamma}_k = f_{\mathbf{D}}^{ODO}(\boldsymbol{\Gamma}_{k-1}, \mathbf{u}_k), \tag{7.23}$$

is the standard equation for the position and orientation evolution with a translation vector and a rotation vector as inputs

$$\begin{bmatrix} \mathbf{t}_k \\ \mathbf{q}_k \end{bmatrix} = \begin{bmatrix} \mathbf{t}_{k-1} + \mathbf{R}\left(\mathbf{q}_{k-1}\right)\Delta\mathbf{t}_k \\ \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix} 0 \\ \Delta\mathbf{r}_k \end{bmatrix} \end{bmatrix}. \tag{7.24}$$

The delta propagation function

$$\delta\boldsymbol{\Gamma}_k = \delta f_{\mathbf{D}}^{ODO}(\boldsymbol{\Gamma}_{k-1}, \delta\boldsymbol{\Gamma}_{k-1}, \mathbf{u}_k, \boldsymbol{\eta}_k), \tag{7.25}$$

results in

$$\begin{bmatrix} \delta\mathbf{t}_k \\ \delta\boldsymbol{\theta}_k \end{bmatrix} = \begin{bmatrix} \delta\mathbf{t}_{k-1} + \mathbf{R}\left(\mathbf{q}_{k-1}\right)\left(-\left[\Delta\mathbf{t}_k\right]_{\times}\delta\boldsymbol{\theta}_{k-1} + \boldsymbol{\eta}_{\mathbf{t}} + \left[\delta\boldsymbol{\theta}_{k-1}\right]_{\times}\boldsymbol{\eta}_{\mathbf{t}}\right) \\ \delta\boldsymbol{\theta}_{k-1} - \left[\mathbf{r}_k + \frac{\boldsymbol{\eta}_{\mathbf{r}}}{2}\right]_{\times}\delta\boldsymbol{\theta}_{k-1} + \boldsymbol{\eta}_{\mathbf{r}} \end{bmatrix}. \tag{7.26}$$

7.2.1.1 Mathematical Derivation

Lets explain in details how to derive the Equation 7.25. Let consider the first order derivative of the robot true pose w.r.t the time $\dot{\tilde{\mathbf{\Gamma}}}$ split in the two terms $\dot{\tilde{\mathbf{t}}}$, $\dot{\tilde{\mathbf{q}}}$. Consider the instantaneous tangential velocity $\tilde{\mathbf{v}} = \mathbf{v} + \boldsymbol{\eta}_{\mathbf{v}}$ and rotational velocity $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}}$:

$$\dot{\tilde{\mathbf{t}}} = \mathbf{R}(\tilde{\mathbf{q}})\,\tilde{\mathbf{v}}, \tag{7.27}$$

$$\dot{\tilde{\mathbf{q}}} = \frac{1}{2}\tilde{\mathbf{q}} \otimes \begin{bmatrix} 0 \\ \tilde{\boldsymbol{\omega}} \end{bmatrix}. \tag{7.28}$$

The previous equations can be rewritten in terms of the nominal values, obtaining

$$\dot{\mathbf{t}} = \mathbf{R}(\mathbf{q})\,\mathbf{v}, \tag{7.29}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{7.30}$$

and using the Eulero method integrations on the Δt interval, we obtain

$$\mathbf{t}_k = \mathbf{t}_{k-1} + \mathbf{R}(\mathbf{q}_{k-1})\,\mathbf{v}_k \Delta t, \tag{7.31}$$

$$\mathbf{q}_k = \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_k \Delta t \end{bmatrix}, \tag{7.32}$$

and by substituting $\mathbf{v}_k \Delta t = \Delta \mathbf{t}_k$ and $\boldsymbol{\omega}_k \Delta t = \Delta \mathbf{r}_k$ the prediction function of Equation 7.24, i.e., for the nominal values, is obtained.

Since $\tilde{\mathbf{t}} = \mathbf{t} + \delta\mathbf{t}$, the time derivative of $\tilde{\mathbf{t}}$ results

$$\dot{\tilde{\mathbf{t}}} = \dot{\mathbf{t}} + \dot{\delta\mathbf{t}}. \tag{7.33}$$

Substituting the Equation 7.27 in the left member of the latter and Equation 7.29 to $\dot{\mathbf{t}}$ in the right member we obtain

$$\mathbf{R}(\tilde{\mathbf{q}})\,\tilde{\mathbf{v}} = \mathbf{R}(\mathbf{q})\,\mathbf{v} + \dot{\delta\mathbf{t}}. \tag{7.34}$$

Now the $\dot{\delta\mathbf{t}}$ can be isolated and the equation expanded thanks to quaternion small rotation approximations

$$\begin{aligned} \dot{\delta\mathbf{t}} &= \mathbf{R}(\tilde{\mathbf{q}})\,\tilde{\mathbf{v}} - \mathbf{R}(\mathbf{q})\,\mathbf{v} && (7.35)\\ &= \mathbf{R}(\mathbf{q})\left[\mathbf{I} + [\delta\boldsymbol{\theta}]_\times + o\left(\|\delta\boldsymbol{\theta}\|^2\right)\right](\mathbf{v} + \boldsymbol{\eta}_{\mathbf{v}}) - \mathbf{R}(\mathbf{q})\,\mathbf{v} && (7.36)\\ &\simeq \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{v} + \mathbf{R}(\mathbf{q})\,\boldsymbol{\eta}_{\mathbf{v}} + \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\boldsymbol{\eta}_{\mathbf{v}} && (7.37)\\ &= -\mathbf{R}(\mathbf{q})\,[\mathbf{v}]_\times\,\delta\boldsymbol{\theta} + \mathbf{R}(\mathbf{q})\,\boldsymbol{\eta}_{\mathbf{v}} + \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\boldsymbol{\eta}_{\mathbf{v}}, && (7.38) \end{aligned}$$

and the time integration in the Δt interval gives

$$\begin{aligned} \delta\mathbf{t}_k &= \delta\mathbf{t}_{k-1} + \Delta t\,\mathbf{R}(\mathbf{q}_{k-1})\left(-[\mathbf{v}_k]_\times\,\delta\boldsymbol{\theta}_{k-1} + \boldsymbol{\eta}_{\mathbf{v}} + [\delta\boldsymbol{\theta}_{k-1}]_\times\,\boldsymbol{\eta}_{\mathbf{v}}\right)\\ &= \delta\mathbf{t}_{k-1} + \mathbf{R}(\mathbf{q}_{k-1})\left(-[\Delta\mathbf{t}_k]_\times\,\delta\boldsymbol{\theta}_{k-1} + \boldsymbol{\eta}_{\mathbf{t}} + [\delta\boldsymbol{\theta}_{k-1}]_\times\,\boldsymbol{\eta}_{\mathbf{t}}\right), \end{aligned} \tag{7.39}$$

being $\boldsymbol{\eta}_{\mathbf{v}} = \Delta t\, \boldsymbol{\eta}_{\mathbf{t}}$. It can be noticed that, apart from the last term, the evolution of the error of the position is linear, since the state elements are $\delta \mathbf{t}_{k-1}$ and $\delta \boldsymbol{\theta}_{k-1}$. The last terms involves both the noise and the state element $\delta \boldsymbol{\theta}_{k-1}$, but we can consider it sufficiently small, since delta values are reset at each step of the EKF.

The prediction of the error state for the orientation terms has to be derived considering the time derivative of the true value of the orientation

$$\dot{\tilde{\mathbf{q}}} = \frac{\partial\left(\mathbf{q} \otimes \delta \mathbf{q}\right)}{\partial t} = \dot{\mathbf{q}} \otimes \delta \mathbf{q} + \mathbf{q} \otimes \dot{\delta \mathbf{q}}. \tag{7.40}$$

Now, substituting Equation 7.28 in the left term and Equation 7.30 in the $\dot{\mathbf{q}}$ element of the right term we obtain

$$\frac{1}{2}\tilde{\mathbf{q}} \otimes \begin{bmatrix} 0 \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \delta \mathbf{q} + \mathbf{q} \otimes \dot{\delta \mathbf{q}}, \tag{7.41}$$

and thanks to the definition of $\tilde{\mathbf{q}} = \mathbf{q} \otimes \delta \mathbf{q}$ we can isolate the $\dot{\delta \mathbf{q}}$ term:

$$\frac{1}{2}\mathbf{q} \otimes \delta \mathbf{q} \otimes \begin{bmatrix} 0 \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \delta \mathbf{q} + \mathbf{q} \otimes \dot{\delta \mathbf{q}}, \tag{7.42}$$

$$\dot{\delta \mathbf{q}} = \frac{1}{2}\delta \mathbf{q} \otimes \begin{bmatrix} 0 \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \delta \mathbf{q}. \tag{7.43}$$

Now we can approximate $\dot{\delta \mathbf{q}}$ with $\begin{bmatrix} 0 \\ \frac{\dot{\delta \boldsymbol{\theta}}}{2} \end{bmatrix}$ and $\delta \mathbf{q}$ with $\begin{bmatrix} 1 \\ \frac{\delta \boldsymbol{\theta}}{2} \end{bmatrix}$ obtaining

$$\begin{aligned}
\begin{bmatrix} 0 \\ \dot{\delta \boldsymbol{\theta}} \end{bmatrix} &= \delta \mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}} \end{bmatrix} - \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \delta \mathbf{q} && (7.44)\\
&= \left[\mathbf{Q}^{+}(\boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}}) - \mathbf{Q}(\boldsymbol{\omega})\right] \delta \mathbf{q} && (7.45)\\
&= \begin{bmatrix} 0 & (-\boldsymbol{\omega} - \boldsymbol{\eta}_{\boldsymbol{\omega}} + \boldsymbol{\omega})^T \\ \boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}} - \boldsymbol{\omega} & -\left[\boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}}\right]_\times - \left[\boldsymbol{\omega}\right]_\times \end{bmatrix} \delta \mathbf{q} && (7.46)\\
&= \begin{bmatrix} 0 & -\boldsymbol{\eta}_{\boldsymbol{\omega}}^T \\ \boldsymbol{\eta}_{\boldsymbol{\omega}} & -\left[2\boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}}\right]_\times \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\delta \boldsymbol{\theta}}{2} \end{bmatrix}, && (7.47)
\end{aligned}$$

that results in the system

$$\begin{cases} 0 = -\boldsymbol{\eta}_{\boldsymbol{\omega}}^T \dfrac{\delta \boldsymbol{\theta}}{2} \\ \dot{\delta \boldsymbol{\theta}} = \boldsymbol{\eta}_{\boldsymbol{\omega}} - \left[2\boldsymbol{\omega} + \boldsymbol{\eta}_{\boldsymbol{\omega}}\right]_\times \dfrac{\delta \boldsymbol{\theta}}{2}. \end{cases} \tag{7.48}$$

The first equation gives no useful informations for the determination of the prediction of the orientation error, while the integration in time of the second vectorial equation gives the prediction of $\delta \boldsymbol{\theta}_k$:

$$\begin{aligned}
\delta \boldsymbol{\theta}_k &= \delta \boldsymbol{\theta}_{k-1} + \Delta t \left(\boldsymbol{\eta}_{\boldsymbol{\omega}} - \left[\boldsymbol{\omega}_k + \frac{\boldsymbol{\eta}_{\boldsymbol{\omega}}}{2}\right]_\times \delta \boldsymbol{\theta}_{k-1}\right) && (7.49)\\
&= \delta \boldsymbol{\theta}_{k-1} - \left[\mathbf{r}_k + \frac{\boldsymbol{\eta}_{\mathbf{r}}}{2}\right]_\times \delta \boldsymbol{\theta}_{k-1} + \boldsymbol{\eta}_{\mathbf{r}}, && (7.50)
\end{aligned}$$

being $\boldsymbol{\eta}_{\mathbf{r}} = \Delta t \boldsymbol{\eta}_{\boldsymbol{\omega}}$.

7.2.2 Motion Model Without Odometric Informations

The motion model in the indirect form when the odometric information is not available, i.e., using a constant velocity motion model, for nominal values is

$$\boldsymbol{\Gamma}_k \quad = \quad f_{\mathbf{D}}^{CSR}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}), \tag{7.51}$$

is the standard equation for the position and orientation prediction, while current tangential and rotational velocity are maintained constant

$$\begin{bmatrix} \mathbf{t}_k \\ \mathbf{q}_k \\ \mathbf{v}_k \\ \boldsymbol{\omega}_k \end{bmatrix} = \begin{bmatrix} \mathbf{t}_{k-1} + \mathbf{R}\left(\mathbf{q}_{k-1}\right) \mathbf{v}_{k-1} \Delta t \\ \mathbf{q}_{k-1} + \frac{1}{2}\mathbf{q}_{k-1} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{k-1}\Delta t \end{bmatrix} \\ \mathbf{v}_{k-1} \\ \boldsymbol{\omega}_{k-1}. \end{bmatrix}. \tag{7.52}$$

The delta propagation function

$$\delta\boldsymbol{\Gamma}_k \quad = \quad \delta f_{\mathbf{D}}^{CSR}(\boldsymbol{\Gamma}_{k-1}, \boldsymbol{\Lambda}_{k-1}, \delta\boldsymbol{\Gamma}_{k-1}, \delta\boldsymbol{\Lambda}_{k-1}, \boldsymbol{\eta}_k), \tag{7.53}$$

results in

$$\begin{bmatrix} \delta\mathbf{t}_k \\ \delta\boldsymbol{\theta}_k \\ \delta\mathbf{v}_k \\ \delta\boldsymbol{\omega}_k \end{bmatrix} = \begin{bmatrix} \delta\mathbf{t}_{k-1} + \Delta t\, \mathbf{R}\left(\mathbf{q}\right) \left(- \left[\mathbf{v}\right]_\times \delta\boldsymbol{\theta} + \delta\mathbf{v} + \left[\delta\boldsymbol{\theta}\right]_\times \delta\mathbf{v}\right) \\ \delta\boldsymbol{\theta}_{k-1} + \Delta t \left(\delta\boldsymbol{\omega} - \left[\boldsymbol{\omega}_k + \frac{\delta\boldsymbol{\omega}}{2}\right]_\times \delta\boldsymbol{\theta}_{k-1}\right) \\ \delta\mathbf{v}_{k-1} + \boldsymbol{\eta}_{\mathbf{v}} \\ \delta\boldsymbol{\omega}_{k-1} + \boldsymbol{\eta}_{\boldsymbol{\omega}} \end{bmatrix}. \tag{7.54}$$

7.2.2.1 Mathematical Derivation

Lets split the true robot pose $\tilde{\boldsymbol{\Gamma}}$ in the true robot position $\tilde{\mathbf{t}}$ and the robot orientation $\tilde{\mathbf{q}}$. Similarly, the motion parameters are expressed by the true tangential velocity in $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}^{(R)}$ expressed in robot frame and the rotational velocity $\tilde{\boldsymbol{\omega}}$.

Alike what done for the motion model based on the odometric information, lets consider the first order derivatives of the true values

$$\dot{\tilde{\mathbf{t}}} \quad = \quad \mathbf{R}\left(\tilde{\mathbf{q}}\right) \tilde{\mathbf{v}}, \tag{7.55}$$

$$\dot{\tilde{\mathbf{q}}} \quad = \quad \frac{1}{2}\tilde{\mathbf{q}} \otimes \begin{bmatrix} 0 \\ \tilde{\boldsymbol{\omega}} \end{bmatrix}, \tag{7.56}$$

$$\dot{\tilde{\mathbf{v}}} \quad = \quad \boldsymbol{\eta}_{\dot{\mathbf{v}}}, \tag{7.57}$$

$$\dot{\tilde{\boldsymbol{\omega}}} \quad = \quad \boldsymbol{\eta}_{\dot{\boldsymbol{\omega}}}, \tag{7.58}$$

and the derivatives of the nominal values

$$\begin{aligned} \dot{\mathbf{t}} &= \mathbf{R}(\mathbf{q})\,\mathbf{v}, && (7.59)\\ \dot{\mathbf{q}} &= \frac{1}{2}\mathbf{q}\otimes\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}, && (7.60)\\ \dot{\mathbf{v}} &= 0, && (7.61)\\ \dot{\boldsymbol{\omega}} &= 0. && (7.62)\end{aligned}$$

The integration, using the Eulero method, of the last equations gives the prediction functions for the nominal values of Equation 7.52.

The time derivative of $\dot{\tilde{\mathbf{t}}}$ is

$$\dot{\tilde{\mathbf{t}}} = \dot{\mathbf{t}} + \dot{\delta\mathbf{t}}, \tag{7.63}$$

and, by substituting Equation 7.55 in the left member of the Equation 7.63 and Equation 7.59 in the $\dot{\mathbf{t}}$ of its right member we obtain:

$$\mathbf{R}(\tilde{\mathbf{q}})\,\tilde{\mathbf{v}} = \mathbf{R}(\mathbf{q})\,\mathbf{v} + \dot{\delta\mathbf{t}}. \tag{7.64}$$

The $\dot{\delta\mathbf{t}}$ term can be isolated and, thanks to the small rotations properties, the following steps can be performed

$$\begin{aligned} \dot{\delta\mathbf{t}} &= \mathbf{R}(\mathbf{q})\left[\mathbf{I} + [\delta\boldsymbol{\theta}]_\times + o\left(\|\delta\boldsymbol{\theta}\|^2\right)\right](\mathbf{v}+\delta\mathbf{v}) - \mathbf{R}(\mathbf{q})\,\mathbf{v} && (7.65)\\ &\simeq \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{v} + \mathbf{R}(\mathbf{q})\,\delta\mathbf{v} + \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\delta\mathbf{v} && (7.66)\\ &= -\mathbf{R}(\mathbf{q})\,[\mathbf{v}]_\times\,\delta\boldsymbol{\theta} + \mathbf{R}(\mathbf{q})\,\delta\mathbf{v} + \mathbf{R}(\mathbf{q})\,[\delta\boldsymbol{\theta}]_\times\,\delta\mathbf{v}. && (7.67)\end{aligned}$$

It can be noticed that only the last term introduces a non linearity, while the others are linear in the state variable $\delta\mathbf{v}$ and $\delta\boldsymbol{\theta}$. The discrete time integration gives the prediction function in the error state formulation for the position term:

$$\delta\mathbf{t}_k = \delta\mathbf{t}_{k-1} + \Delta t\,\mathbf{R}(\mathbf{q})\left(-[\mathbf{v}]_\times\,\delta\boldsymbol{\theta} + \delta\mathbf{v} + [\delta\boldsymbol{\theta}]_\times\,\delta\mathbf{v}\right). \tag{7.68}$$

Lets now consider the time derivative of the orientation

$$\dot{\tilde{\mathbf{q}}} = \frac{\partial\mathbf{q}\otimes\delta\mathbf{q}}{\partial t} = \dot{\mathbf{q}}\otimes\delta\mathbf{q} + \mathbf{q}\otimes\dot{\delta\mathbf{q}}. \tag{7.69}$$

By substituting Equation 7.56 in the left member and Equation 7.60 in the $\dot{\mathbf{q}}$ of the right member we obtain

$$\begin{aligned} \frac{1}{2}\tilde{\mathbf{q}}\otimes\begin{bmatrix}0\\ \tilde{\boldsymbol{\omega}}\end{bmatrix} &= \dot{\mathbf{q}}\otimes\delta\mathbf{q} + \mathbf{q}\otimes\dot{\delta\mathbf{q}}, && (7.70)\\ \frac{1}{2}\mathbf{q}\otimes\delta\mathbf{q}\otimes\begin{bmatrix}0\\ \boldsymbol{\omega}+\delta\boldsymbol{\omega}\end{bmatrix} &= \frac{1}{2}\mathbf{q}\otimes\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}\otimes\delta\mathbf{q} + \mathbf{q}\otimes\dot{\delta\mathbf{q}}. && (7.71)\end{aligned}$$

The term $\dot{\delta \mathbf{q}}$ can be isolated and $\delta \mathbf{q}$ and $\dot{\delta \mathbf{q}}$ can be respectively approximated with $\begin{bmatrix} 1 \\ \frac{\delta \boldsymbol{\theta}}{2} \end{bmatrix}$ and $\begin{bmatrix} 0 \\ \frac{\dot{\delta \boldsymbol{\theta}}}{2} \end{bmatrix}$

$$\begin{aligned}
\dot{\delta \mathbf{q}} &= \frac{1}{2}\delta \mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} + \delta\boldsymbol{\omega} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \delta \mathbf{q}, && (7.72) \\
\begin{bmatrix} 0 \\ \dot{\delta \boldsymbol{\theta}} \end{bmatrix} &= \left[\mathbf{Q}^{+}\left(\boldsymbol{\omega} + \delta\boldsymbol{\omega}\right) - \mathbf{Q}\left(\boldsymbol{\omega}\right)\right] \delta \mathbf{q} && (7.73) \\
&= \begin{bmatrix} 0 & \left(-\boldsymbol{\omega} - \delta\boldsymbol{\omega} + \boldsymbol{\omega}\right)^{T} \\ \left(\boldsymbol{\omega} + \delta\boldsymbol{\omega} - \boldsymbol{\omega}\right) & -\left[\boldsymbol{\omega} + \delta\boldsymbol{\omega}\right]_{\times} - \left[\boldsymbol{\omega}\right]_{\times} \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\delta \boldsymbol{\theta}}{2} \end{bmatrix} && (7.74) \\
&= \begin{bmatrix} 0 & -\delta\boldsymbol{\omega}^{T} \\ \delta\boldsymbol{\omega} & -\left[2\boldsymbol{\omega} + \delta\boldsymbol{\omega}\right]_{\times} \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\delta \boldsymbol{\theta}}{2} \end{bmatrix}, && (7.75)
\end{aligned}$$

that results in the system

$$\begin{cases} 0 = -\delta\boldsymbol{\omega}^{T} \dfrac{\delta \boldsymbol{\theta}}{2} \\ \dot{\delta \boldsymbol{\theta}} = \delta\boldsymbol{\omega} - \left[2\boldsymbol{\omega} + \delta\boldsymbol{\omega}\right]_{\times} \dfrac{\delta \boldsymbol{\theta}}{2}. \end{cases} \tag{7.76}$$

The first equation gives no useful informations for the determination of the orientation prediction term, while the integration in time of the second vectorial equation gives the prediction of $\delta \boldsymbol{\theta}_k$:

$$\delta \boldsymbol{\theta}_k = \delta \boldsymbol{\theta}_{k-1} + \Delta t \left(\delta\boldsymbol{\omega} - \left[\boldsymbol{\omega}_k + \frac{\delta\boldsymbol{\omega}}{2}\right]_{\times} \delta \boldsymbol{\theta}_{k-1} \right) \tag{7.77}$$

Consider now the tangential velocity term $\tilde{\mathbf{v}}$. Its derivate is $\dot{\tilde{\mathbf{v}}} = \dot{\mathbf{v}} + \dot{\delta \mathbf{v}}$. The left member can be substituted by Equation 7.57, while $\dot{\mathbf{v}}$ in the right member is zero thanks to Equation 7.61. This implies that $\dot{\delta \mathbf{v}} = \boldsymbol{\eta}_{\dot{\mathbf{v}}}$, thus $\delta \mathbf{v}_k = \delta \mathbf{v}_{k-1} + \boldsymbol{\eta}_{\mathbf{v}}$, being $\Delta t\, \boldsymbol{\eta}_{\dot{\mathbf{v}}} = \boldsymbol{\eta}_{\mathbf{v}}$. The same reasoning leads to the prediction equation of the error state for the rotational velocity.

We do not treat the case of the motion model without odometric information with the velocity expressed in the world reference system; it can be derived by following the same procedure sketched in this section.

7.3 Parameterization

Hereafter two of the parameterizations proposed in the previous chapter are modified to be used in the Hybrid Indirect EKF-SLAM. The chosen parameterizations are Framed Homogeneous Point and the Framed Inverse Depth, i.e., the two parameterizations with the anchor frame. The use of the indirect EKF formulation is very helpful for these parameterizations for two reasons. Firstly, the presence of the quaternion in the anchor frame makes unusable the Conditional Independent SLAM framework, since the covariance matrix becomes singular when a quaternion is present. Secondly, the use of the indirect formulation allows to insert a rotation vector, which dimension is 3, instead of a quaternion, which dimension is 4 in the EKF vector state. This implies that the dimension of the anchor frame decreases from 7 to 6 elements, reducing the complexity a bit further.

7.3.1 Anchor Frame

The anchor frame is represented by its nominal part translation vector $\mathbf{t}$ and quaternion $\mathbf{q}$, and by its error-state part: the translation delta vector $\delta\mathbf{t}$ and the rotation vector $\delta\boldsymbol{\theta}$. The true translation $\tilde{\mathbf{t}}$ and the true rotation quaternion $\tilde{\mathbf{q}}$ are respectively given by $\mathbf{t} + \delta\mathbf{t}$ and by $\tilde{\mathbf{q}} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}$.

7.3.1.1 Anchor Frame Initialization

The initialization of an anchor frame has to be specified both for the nominal part and for the error state part. Given the current nominal robot pose $\boldsymbol{\Gamma}_k = [\mathbf{t}_k^T, \mathbf{q}_k^T]^T$, the delta position $\delta\boldsymbol{\Gamma}_k = [\delta\mathbf{t}_k^T, \delta\boldsymbol{\theta}_k]^T$ and the transformation between the robot and the camera $\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}$, the function that initialize the new nominal part of the anchor frame is

$$\mathbf{y}_{new}^{AF_i} = g^{AF_i}(\boldsymbol{\Gamma}_k, \mathbf{s} = \{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\}) = \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \mathbf{t}_k \\ \mathbf{q}_k \otimes \mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}} \end{bmatrix}, \tag{7.78}$$

while the function that initialize the delta elements is

$$\delta\mathbf{y}_{new}^{AF_i} = \delta g^{AF_i}(\boldsymbol{\Gamma}_k, \delta\boldsymbol{\Gamma}_k, \mathbf{s} = \{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\}) = \begin{bmatrix} \delta\mathbf{t}_k - \mathbf{R}(\mathbf{q}_k)\left[\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right]_\times \delta\boldsymbol{\theta}_k \\ \frac{\mathbf{q}_{C\,w}^2\,\delta\boldsymbol{\theta}_k + 2\,\mathbf{q}_{C\,w}\,[\delta\boldsymbol{\theta}_k]_\times \mathbf{q}_{C\,\boldsymbol{\theta}} - [\mathbf{q}_{C\,\boldsymbol{\theta}}]_\times [\mathbf{q}_{C\,\boldsymbol{\theta}}]_\times \delta\boldsymbol{\theta}_k}{1 + \mathbf{q}_{C\,\boldsymbol{\theta}}^T \left[\frac{\delta\boldsymbol{\theta}_k}{2}\right]_\times \mathbf{q}_{C\,\boldsymbol{\theta}}} \end{bmatrix}, \tag{7.79}$$

where the rotation quaternion of the camera w.r.t the robot frame has been decomposed as $\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}} = \left[\mathbf{q}_{C\,w}, \mathbf{q}_{C\,\boldsymbol{\theta}}^T\right]^T$. Notice that we indicate with the subscript i the indirect formulation of the anchor frame.

7.3.1.2 Mathematical Derivation

The nominal part of the anchor frame is derived by the standard frame composition rules applied to the nominal values of the translation and rotation. By considering the same equation on the true value it is possible to express the new true frame, composed by $\tilde{\mathbf{t}}$ and $\tilde{\mathbf{q}}$ as

$$\tilde{\mathbf{t}} = \mathbf{R}(\tilde{\mathbf{q}}_k)\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}} + \tilde{\mathbf{t}}_k, \tag{7.80}$$

$$\tilde{\mathbf{q}} = \tilde{\mathbf{q}}_k \otimes \mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}}. \tag{7.81}$$

The left members can be substituted by the definition of the frame true value, while the right members can be expanded with the composition of the robot pose nominal and delta values:

$$\mathbf{t} + \delta\mathbf{t} = \mathbf{R}(\mathbf{q}_k)\left[\mathbf{I} + [\delta\boldsymbol{\theta}_k]_\times + o\left(\|\delta\boldsymbol{\theta}_k\|^2\right)\right]\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \mathbf{t}_k + \delta\mathbf{t}_k, \tag{7.82}$$

$$\mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} = \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}. \tag{7.83}$$

In the first equation it is possible to substitute the term $\mathbf{t}$ in the left member with the first part of the nominal vector of Equation 7.78 and, after discarding the second order terms, it results

$$\delta\mathbf{t} \simeq \mathbf{R}(\mathbf{q}_k)\,[\delta\boldsymbol{\theta}_k]_\times\,\mathbf{t}^{\mathcal{R}}_{\mathcal{C}_i} + \delta\mathbf{t}_k, \tag{7.84}$$

which is equivalent to the first part of the delta anchor frame initialization expressed in Equation 7.79.

Similarly, by substituting the $\mathbf{q}$ term with the second part of the nominal anchor frame of Equation 7.78, we obtain

$$\mathbf{q}_k \otimes \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} = \eta\,\mathbf{q}_k \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}, \tag{7.85}$$

where η is a normalization factor. By isolating the term $\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}$ in the left member we obtain

$$\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} = \eta\,\mathbf{q}^{\mathcal{R}\,*}_{\mathcal{C}_i} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}. \tag{7.86}$$

The quaternion composition can be expressed with the $\mathbf{Q}(\cdot)$ and $\mathbf{Q}^+(\cdot)$ matrix, resulting in

$$\begin{aligned}
\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix} &= \eta\,\mathbf{Q}\left(\mathbf{q}^{\mathcal{R}\,*}_{\mathcal{C}_i}\right)\mathbf{Q}^+\left(\mathbf{q}^{\mathcal{R}}_{\mathcal{C}_i}\right) \\
&= \eta \begin{bmatrix} \mathbf{q}_{C_w} & \mathbf{q}_{C_{\boldsymbol{\theta}}}^T \\ -\mathbf{q}_{C_{\boldsymbol{\theta}}} & \mathbf{q}_{C_w}\mathbf{I} - [\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times \end{bmatrix} \begin{bmatrix} \mathbf{q}_{C_w} & -\mathbf{q}_{C_{\boldsymbol{\theta}}}^T \\ \mathbf{q}_{C_{\boldsymbol{\theta}}} & \mathbf{q}_{C_w}\mathbf{I} - [\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \\
&= \eta \begin{bmatrix} \mathbf{q}_{C_w}^2 + \mathbf{q}_{C_{\boldsymbol{\theta}}}^T\mathbf{q}_{C_{\boldsymbol{\theta}}} - \mathbf{q}_{C_{\boldsymbol{\theta}}}^T[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times \frac{\delta\boldsymbol{\theta}_k}{2} \\ -[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times\mathbf{q}_{C_{\boldsymbol{\theta}}} + \left(\mathbf{q}_{C_w}^2 - 2\mathbf{q}_{C_w}[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times + \mathbf{q}_{C_{\boldsymbol{\theta}}}\mathbf{q}_{C_{\boldsymbol{\theta}}}^T + [\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times\right)\frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix}.
\end{aligned} \tag{7.87}$$

The first line allows to derive the η normalization factor:

$$\eta = \frac{1}{\mathbf{q}_{C_w}^2 + \mathbf{q}_{C_{\boldsymbol{\theta}}}^T\mathbf{q}_{C_{\boldsymbol{\theta}}} - \mathbf{q}_{C_{\boldsymbol{\theta}}}^T[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times \frac{\delta\boldsymbol{\theta}_k}{2}} = \frac{1}{1 + \mathbf{q}_{C_{\boldsymbol{\theta}}}^T\left[\frac{\delta\boldsymbol{\theta}_k}{2}\right]_\times\mathbf{q}_{C_{\boldsymbol{\theta}}}}, \tag{7.88}$$

where $\mathbf{q}_{C_w}^2 + \mathbf{q}_{C_{\boldsymbol{\theta}}}^T\mathbf{q}_{C_{\boldsymbol{\theta}}} = 1$ since the camera orientation w.r.t. the robot reference frame is a unit quaternion. Once the η normalization factor has been expressed, the second part of Equation 7.87 allows to compute the orientation part of the error state frame:

$$\delta\boldsymbol{\theta} = \eta\left(\mathbf{q}_{C_w}^2 - 2\mathbf{q}_{C_w}[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times + \mathbf{q}_{C_{\boldsymbol{\theta}}}\mathbf{q}_{C_{\boldsymbol{\theta}}}^T + [\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times\right)\delta\boldsymbol{\theta}_k, \tag{7.89}$$

where it has to be noticed that the term $[\mathbf{q}_{C_{\boldsymbol{\theta}}}]_\times\mathbf{q}_{C_{\boldsymbol{\theta}}}$ is zero.

7.3.1.3 Frame Transformation

When used in the CI SLAM framework, in order to perform the map transition step, the anchor frame needs to be transformed in the reference system of the new map, i.e., it has to be composed with the inverse transformation that describe the robot pose in the current map. This has to be done both for the nominal and the error state part: given the anchor frame $\mathbf{y}^{AF_i} = \left[\mathbf{t}^T, \mathbf{q}^T\right]^T$, the nominal part of the transformed anchor frame is given by

$$\begin{bmatrix} \mathbf{t}^+ \\ \mathbf{q}^+ \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{q}_k)^T (\mathbf{t} - \mathbf{t}_k) \\ \mathbf{q}_k^* \otimes \mathbf{q} \end{bmatrix}, \tag{7.90}$$

where $\mathbf{t}_k$ and $\mathbf{q}_k$ are as usual the robot position and orientation in the current map. The new error state part of the frame has to be calculated by $\mathbf{y}^{AF_i}$ and its error state part $\delta\mathbf{y}^{AF_i}$ as

$$\begin{bmatrix} \delta\mathbf{t}^+ \\ \delta\boldsymbol{\theta}^+ \end{bmatrix} = \begin{bmatrix} \left(\mathbf{I} - [\delta\boldsymbol{\theta}]_\times\right) \mathbf{R}\left(\mathbf{q_k}\right)^T (\delta\mathbf{t} - \delta\mathbf{t}_k) - [\delta\boldsymbol{\theta}]_\times \mathbf{R}\left(\mathbf{q_k}\right)^T (\mathbf{t} - \mathbf{t}_k) \\ 2\eta\check{\mathbf{q}}_{\boldsymbol{\theta}} \end{bmatrix}, \tag{7.91}$$

where $\check{\mathbf{q}}_{\boldsymbol{\theta}}$ are last three elements, i.e., the axis part, of the quaternion

$$\check{\mathbf{q}} = \mathbf{q}^* \otimes \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ -\frac{\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}_k^* \otimes \mathbf{q} \otimes \begin{bmatrix} 1 \\ -\frac{\boldsymbol{\theta}}{2} \end{bmatrix}, \tag{7.92}$$

and η is given by the inverse of the scalar element of the quaternion $\check{\mathbf{q}}$

$$\eta = \frac{1}{\check{\mathbf{q}}_w}. \tag{7.93}$$

7.3.1.4 Mathematical Derivation

Lets consider the current true robot position $\tilde{\mathbf{t}}_k = \mathbf{t}_k + \delta\mathbf{t}_k$ and orientation $\tilde{\mathbf{q}}_k = \mathbf{q}_k \otimes \delta\mathbf{q}_k = \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix}$ and the anchor frame expressed by $\tilde{\mathbf{t}} = \mathbf{t} + \delta\mathbf{t}$ and $\tilde{\mathbf{q}} = \mathbf{q} \otimes \delta\mathbf{q} = \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}$. The inverse composition of the true current robot pose and the true anchor frame gives

$$\begin{bmatrix} \tilde{\mathbf{p}}^+ \\ \tilde{\mathbf{q}}^+ \end{bmatrix} = \begin{bmatrix} \mathbf{R}\left(\tilde{\mathbf{q}_k}\right)^T \left(\tilde{\mathbf{t}} - \tilde{\mathbf{t}}_k\right) \\ \tilde{\mathbf{q}}_k^* \otimes \tilde{\mathbf{q}} \end{bmatrix}. \tag{7.94}$$

The expansion of the left and the right members of the first equation gives, discarding the terms with order greater than 1:

$$\tilde{\mathbf{t}}^+ + \delta\tilde{\mathbf{t}}^+ \simeq \left[\mathbf{R}\left(\mathbf{q}_k\right)\left(\mathbf{I} + [\delta\boldsymbol{\theta}_k]_\times\right)\right]^T (\mathbf{t} + \delta\mathbf{t} - \mathbf{t}_k - \delta\mathbf{t}_k), \tag{7.95}$$

then, considering the nominal part of the transformed anchor frame $\mathbf{t}^+ = \mathbf{R}\left(\mathbf{q}_k\right)^T (\mathbf{t} - \mathbf{t}_k)$ we obtain

$$\begin{aligned} \delta\tilde{\mathbf{t}}^+ &\simeq \left[\mathbf{R}\left(\mathbf{q}_k\right)\left(\mathbf{I} + [\delta\boldsymbol{\theta}_k]_\times\right)\right]^T (\delta\mathbf{t} - \delta\mathbf{t}_k) + [\delta\boldsymbol{\theta}_k]_\times^T \mathbf{R}\left(\mathbf{q}_k\right)^T (\mathbf{t} - \mathbf{t}_k) \quad (7.96) \\ &= \left(\mathbf{I} - [\delta\boldsymbol{\theta}_k]_\times\right) \mathbf{R}\left(\mathbf{q}_k\right)^T (\delta\mathbf{t} - \delta\mathbf{t}_k) - [\delta\boldsymbol{\theta}_k]_\times \mathbf{R}\left(\mathbf{q}_k\right)^T (\mathbf{t} - \mathbf{t}_k). \quad (7.97) \end{aligned}$$

The expansion of the left and right member of the second part of the Equation 7.94 gives

$$\mathbf{q}^{+} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}^{+}}{2} \end{bmatrix} = \eta \begin{bmatrix} 1 \\ -\frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}_k^{*} \otimes \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}, \tag{7.98}$$

where the η term was inserted to normalize the members. The nominal part of the left member $\mathbf{q}^{+}$ can be substituted with $\mathbf{q}_k^{*} \otimes \mathbf{q}$ and can be moved to the right terms, giving

$$\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}^{+}}{2} \end{bmatrix} = \eta\, \mathbf{q}^{*} \otimes \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ -\frac{\delta\boldsymbol{\theta}_k}{2} \end{bmatrix} \otimes \mathbf{q}_k^{*} \otimes \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2} \end{bmatrix}. \tag{7.99}$$

Naming $\check{\mathbf{q}}$ the right member but η, gives the possibility to write

$$\begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}^{+}}{2} \end{bmatrix} = \eta \begin{bmatrix} \check{\mathbf{q}}_w \\ \check{\mathbf{q}}_{\boldsymbol{\theta}} \end{bmatrix}, \tag{7.100}$$

then it is possible to compute η from the first line and $\delta\boldsymbol{\theta}^{+}$ from the last part of the vectors.

7.3.2 Indirect Framed Homogeneous Point - FHP$_i$

The extension of the framed homogeneous point parameterization to the indirect formulation is simple: the true value of the landmark $\tilde{\mathbf{y}}^{FHP_i} = \begin{bmatrix} \tilde{\boldsymbol{p}_\pi}^T & \tilde{w} \end{bmatrix}^T$ is simply given by the sum of the nominal and delta part

$$\tilde{\mathbf{y}}^{FHP_i} = \mathbf{y}^{FHP_i} + \delta\mathbf{y}^{FHP_i}, \tag{7.101}$$

$$\begin{bmatrix} \tilde{\boldsymbol{p}_\pi} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_\pi \\ w \end{bmatrix} + \begin{bmatrix} \delta\boldsymbol{p}_\pi \\ \delta w \end{bmatrix}. \tag{7.102}$$

7.3.2.1 Euclidean Point

A FHP landmark, composed by its nominal $\mathbf{y}^{FHP_i}$ and delta values $\delta\mathbf{y}^{FHP_i}$, applied to its anchor frame $\mathbf{y}^{AF_i}$, $\delta\mathbf{y}^{AF_i}$, correspond to the 3D Euclidean point

$$\mathbf{Y} = \tilde{p}^{FHP_i}\left(\mathbf{y}^{AF_i}, \delta\mathbf{y}^{AF_i}, \mathbf{y}^{FHP_i}, \delta\mathbf{y}^{FHP_i}\right) = \tag{7.103}$$

$$= \mathbf{t} + \delta\mathbf{t} + \frac{1}{w + \delta w}\mathbf{R}\left(\mathbf{q}\right)\left[\mathbf{I} + \left[\delta\boldsymbol{\theta}\right]_\times\right] \begin{bmatrix} \boldsymbol{p}_\pi + \delta\boldsymbol{p}_\pi \\ 1 \end{bmatrix}. \tag{7.104}$$

7.3.2.2 FHP Initialization

The initialization function has to be split in the nominal part and in the delta part as:

$$\mathbf{y}_{new}^{FHP_{i\pi}} = g^{FHP_{i\pi}}\left(\mathbf{s} = \left\{\boldsymbol{p}_{img}, w_0\right\}\right) = \begin{bmatrix} K^I\left(\boldsymbol{p}_{img}\right)^{-1} \\ w_0 \end{bmatrix} \tag{7.105}$$

$$\delta\mathbf{y}_{new}^{FHP_{i\pi}} = \delta g^{FHP_{i\pi}}\left(\mathbf{s} = \left\{\boldsymbol{p}_{img}, w_0\right\}, \boldsymbol{\xi}\right) = \tag{7.106}$$

$$= \begin{bmatrix} K^I\left(\tilde{\boldsymbol{p}}_{img}\right)^{-1} - K^I\left(\boldsymbol{p}_{img}\right)^{-1} \\ \boldsymbol{\xi}_w \end{bmatrix} \tag{7.107}$$

$$\tag{7.108}$$

The $\delta\mathbf{y}_{new}^{FHP_{i\pi}}$ is obtained by subtracting the nominal values $\mathbf{y}_{new}^{FHP_{i\pi}}$ from the values $\tilde{\mathbf{y}}_{new}^{FHP_{i\pi}}$, defined as

$$\tilde{\mathbf{y}}_{new}^{FHP_{i\pi}} = \begin{bmatrix} K^I\left(\tilde{\boldsymbol{p}}_{img}\right)^{-1} \\ w_0 + \boldsymbol{\xi}_w \end{bmatrix}. \tag{7.109}$$

The computation of the mean values of $\delta\mathbf{y}_{new}^{FHP_\pi}$ results in a null vector, but the formula was explicitly written in this form to underline that the Jacobians w.r.t the input noises are not null, as results from the next equation

$$\frac{\partial\delta g^{FHP_{i\pi}}(\cdots)}{\partial\boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial K^I\left(\tilde{\boldsymbol{p}}_{img}\right)^{-1}}{\boldsymbol{\xi}_{p_{img}}} & 0 \\ 0 & 1 \end{bmatrix}. \tag{7.110}$$

7.3.2.3 Measurement Function

The measurement function of a FHP$_i$ landmark $\mathbf{y}_k^{FHP_i}$, $\delta\mathbf{y}_k^{FHP_i}$ involves the current robot pose $\boldsymbol{\Gamma}_k$, $\delta\boldsymbol{\Gamma}_k$, the anchor frame $\mathbf{y}_k^{AF_i}$ and $\delta\mathbf{y}_k^{AF_i}$ to obtain

$$\begin{aligned}&\tilde{h}^{FHP_i}(\boldsymbol{\Gamma}_k, \delta\boldsymbol{\Gamma}_k, \mathbf{y}_k^{AF_i}, \delta\mathbf{y}_k^{AF_i}, \mathbf{y}_k^{FHP_i}, \delta\mathbf{y}_k^{FHP_i}, \mathbf{v} = \left\{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\right\}, \boldsymbol{\delta}) = \\ &= K_i\left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{R}\left(\tilde{\mathbf{q}}_k\right)^T\left(\tilde{w}\left[\tilde{\mathbf{t}} - \tilde{\mathbf{t}}_k\right] + \mathbf{R}\left(\tilde{\mathbf{q}}\right)\begin{bmatrix}\tilde{\boldsymbol{p}}_\pi \\ 1\end{bmatrix}\right) - \tilde{w}\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right) + \boldsymbol{\delta}.\end{aligned} \tag{7.111}$$

where the elements indicated with the tilde have to be computed trough composition of their nominal values and delta values (e.g., $\tilde{\mathbf{t}}_k = \mathbf{t}_k + \delta\mathbf{t}_k$, $\tilde{\mathbf{q}}_k = \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ \frac{\delta\boldsymbol{\theta}}{2}\end{bmatrix}$, etc.).

As stated for the standard formulation of the FHP parameterization, the landmark are expressed in the camera reference system stored in the corresponding anchor frame. This implies that also in the indirect formulation the proper landmark part has only to be shared with the new map, without any transformation. Obviously both the nominal and the delta part has to be shared and properly copied.

7.3.3 Indirect Framed Inverse Scale - FIS$_i$

The FIS parameterization in its indirect formulation is given by the linear summation of the inverse depth nominal value and delta value, thus the true value $\tilde{\mathbf{y}}^{FIS_i} = \left[\tilde{w}\right]$ is given by

$$\begin{aligned}\tilde{\mathbf{y}}^{FIS_i} &= \mathbf{y}^{FIS_i} + \delta\mathbf{y}^{FIS_i}, & (7.112)\\ \tilde{w} &= w + \delta w. & (7.113)\end{aligned}$$

7.3.3.1 Euclidean Point

A FIS$_i$ landmark, composed by its nominal inverse scale $\mathbf{y}^{FIS_i}$ and delta inverse scale $\delta\mathbf{y}^{FIS_i}$, applied to its anchor frame $\mathbf{y}^{AF_i}$, $\delta\mathbf{y}^{AF_i}$ and accompanied by the image point in which the landmark was firstly perceived $\boldsymbol{p}_{img_0}$, correspond to the 3D Euclidean point

$$\begin{aligned}\mathbf{Y} &= \tilde{p}^{FIS_i}\left(\mathbf{y}^{AF_i}, \delta\mathbf{y}^{AF_i}, \mathbf{y}^{FIS_i}, \delta\mathbf{y}^{FIS_i}, \mathbf{s} = \boldsymbol{p}_{img_0}, \boldsymbol{\xi}\right) = & (7.114)\\ &= \mathbf{t} + \delta\mathbf{t} + \frac{1}{w + \delta w}\mathbf{R}(\mathbf{q})\left[\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right]\tilde{\mathbf{r}}_0^{(\mathcal{C})}, & (7.115)\end{aligned}$$

where

$$\begin{aligned}\tilde{\mathbf{r}}_0^{(\mathcal{C})} &= K^{-1}(\tilde{\boldsymbol{p}}_{img_0}), & (7.116)\\ \tilde{\boldsymbol{p}}_{img_0} &= \boldsymbol{p}_{img_0} + \boldsymbol{\xi}, & (7.117)\end{aligned}$$

and the additional $\boldsymbol{\xi}$ noise is introduced to compensate the initial uncertainty on the point that is not estimated during the estimation.

7.3.3.2 Initialization

The initialization of the FIS$_i$ parameterization has only to define the nominal and delta value of the inverse depth, thus

$$\begin{aligned}\mathbf{y}_{new}^{FIS_{i\pi}} &= g^{FIS_{i\pi}}(\mathbf{s} = w_0) = w_0, & (7.118)\\ \delta\mathbf{y}_{new}^{FIS_{i\pi}} &= \delta g^{FIS_{i\pi}}(\boldsymbol{\xi}) = \boldsymbol{\xi}_w; & (7.119)\end{aligned}$$

this comes from $\tilde{w} = w + \delta w = w_0 + \boldsymbol{\xi}_w$. The initial point $\boldsymbol{p}_{img_0}$ has to be stored for the successive measurements of the point.

7.3.3.3 Measurement Function

The measurement function of a FIS$_i$ landmark $\mathbf{y}_k^{FIS_i}$, $\delta\mathbf{y}_k^{FIS_i}$ involves the current robot pose $\boldsymbol{\Gamma}_k$, $\delta\boldsymbol{\Gamma}_k$, the anchor frame $\mathbf{y}_k^{AF_i}$ and $\delta\mathbf{y}_k^{AF_i}$ to obtain

$$\begin{aligned}&\tilde{h}^{FIS_i}(\boldsymbol{\Gamma}_k, \delta\boldsymbol{\Gamma}_k, \mathbf{y}_k^{AF_i}, \delta\mathbf{y}_k^{AF_i}, \mathbf{y}_k^{FIS_i}, \delta\mathbf{y}_k^{FIS_i}, \mathbf{v} = \left\{\mathbf{T}_{\mathcal{C}_i}^{\mathcal{R}}\right\}, \boldsymbol{\delta}) =\\ &= K_i\left(\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{R}(\tilde{\mathbf{q}}_k)^T\left(\tilde{w}\left[\tilde{\mathbf{t}} - \tilde{\mathbf{t}}_k\right] + \mathbf{R}(\tilde{\mathbf{q}})\tilde{\mathbf{r}}_0^{(\mathcal{C})}\right) - \tilde{w}\mathbf{R}(\mathbf{q}_{\mathcal{C}_i}^{\mathcal{R}})^T\mathbf{t}_{\mathcal{C}_i}^{\mathcal{R}}\right) + \boldsymbol{\delta},\end{aligned} \qquad (7.120)$$

where the elements indicated with the tilde have to be computed trough composition of their nominal values and delta values (e.g., $\tilde{\mathbf{t}}_k = \mathbf{t}_k + \delta\mathbf{t}_k$, $\tilde{\mathbf{q}}_k = \mathbf{q}_k \otimes \begin{bmatrix}1\\ \frac{\delta\boldsymbol{\theta}}{2}\end{bmatrix}$, etc.), apart from the viewing ray in camera coordinate $\tilde{\mathbf{r}}_0^{(\mathcal{C})}$ that is

$$\begin{aligned}\tilde{\mathbf{r}}_0^{(\mathcal{C})} &= K^{-1}(\tilde{\boldsymbol{p}}_{img_0}), & (7.121)\\ \tilde{\boldsymbol{p}}_{img_0} &= \boldsymbol{p}_{img_0} + \boldsymbol{\delta}_{34}, & (7.122)\end{aligned}$$

where, as in the standard formulation, we provide an additional noise $\boldsymbol{\delta}_{34}$ to model the uncertainty on the initial point.

As for the FHP$_i$ parameterization, FIS$_i$ landmarks need only to be copied and shared with the new map at the map transition step of the CI-SLAM framework.

CHAPTER 8

CIBA-SLAM: Conditional Independent Bundle Adjusted SLAM

The SLAM problem plays a key role towards the complete autonomy of mobile robots. Consequently, a strong requirement for a practical SLAM implementation is to operate in real time on large environments. The EKF SLAM algorithm represents one of the most studied and well-founded solution to the SLAM problem, but it suffer two important limitations: firstly, it can not handle large environments without losing the ability of operate in real time; secondly, the inconsistency caused by the linearizations performed in the EKF limits the solution reliability and precision. The CI-SLAM framework solves the issue of large maps by partitioning the problem in several local submaps without any information loss, bounding the computational cost of a single map to $O(1)$ and providing procedures to update the complete solution and perform loop closures. Moreover, the use of local maps limits the loose of consistency, since each map is started in a local origin with zero uncertainty. Consequently, with a vision that tends to provide a system more close as possible to a deployment on a real robot, we consider the CI-SLAM framework a good choice in the panorama of the SLAM techniques, although it does not reach the precision of the method based on non linear optimization, also known as Bundle Adjustment. A different aspect of the CI-SLAM framework, that has remained hidden up to now, is that it can be considered as an additional module of an EKF SLAM system, since it adds additional functionalities without modifying the underlaying mechanism. In terms of software engineering it can be though as an *add-on* or *plug-in* to the EKF SLAM basic technique that can be "installed" on the system when needed, otherwise it can be easily avoided.

Algorithm 8 $\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \text{CIBA-SLAM}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

```
1: $[\mathcal{G}, \mathcal{E}] = \text{EKF2Graph}\,(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
2: $\mathcal{G}^* = \text{optimize\_graph}\,(\mathcal{G}, \mathcal{E})$
3: $\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \text{Graph2EKF}\,(\mathcal{G}^*, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
```

In this chapter we ask if it is possible to develop a further module that acts as a plug-in to the CI-SLAM and solves, at least partially, the issues related to linearizations. Such module is developed as a Non Linear Lest Squares optimization (i.e., Bundle Adjustment) applied to each local map of the CI-SLAM system, performed at each map transition step and propagated to the source map, maintaining unchanged the CI-SLAM framework structure. We use the abbreviation *CIBA-SLAM* (Conditionally Independent Bundle Adjusted SLAM) for the proposed method. The proposed module acts as an *add-on* to the SLAM system and in a real application we can choose to activate it if the computational resources of the running system are sufficient, otherwise it can be just turned off. Section 8.1 explains how information are transfered from a Gaussian submap of the CI-SLAM system to an optimization graph, in Section 8.2 the graph structure is formalized in terms of nodes and edges and Section 8.3 explains how the graph optimization solution is propagated back to the original Gaussian submap, completing the operation performed by the CIBA-SLAM plugin. The operations performed by the CIBA-SLAM plugin are summarized in Algorithm 8, which has to be invoked at each map transition step of the CI-SLAM framework. The algorithm has three step: the first converts the Gaussian map represented in the EKF state into a graph, the second optimizes the graph and the third converts back the optimized graph to the Gaussian representation of the EKF.

8.1 From Gaussian Local Map To Graph Optimization

In Chapter 5 the SLAM problem has been presented in two different graphical formulations: as a Dynamic Bayesian Network (DBN) and as an optimization problem on a Graph. The first was solved in an on-line fashion with the EKF-SLAM approach, while the second can be solved as a batch optimization problem with Bundle Adjustment (or Non Linear Least Squares techniques).

In the EKF on-line formulation, past poses of the robot and past measurements are marginalized out, thanks to the Markov property. Unfortunately, the presence of non linear functions in the mathematical model of the system introduces approximations which results in a non optimal solution computed with the EKF mechanism. Despite this, the provided solution is a good starting point for the computation of a better estimation using a Non Linear Least Square technique. A straightforward method can be easily developed by initializing an optimization graph with the robot pose estimates collected in time and the landmarks converted in Euclidean 3D points. This approach is used in [95], where the landmarks are represented with the UID parameterization in an Information Filter and converted to 3D points to initialize a Bundle Adjustment system. It can be noticed that using this approach, the landmarks are estimated with the complete set of available measurements, while estimates of robot poses are simply collected during time, since they are marginalized out from the filter state at each prediction step. This implies that past robot

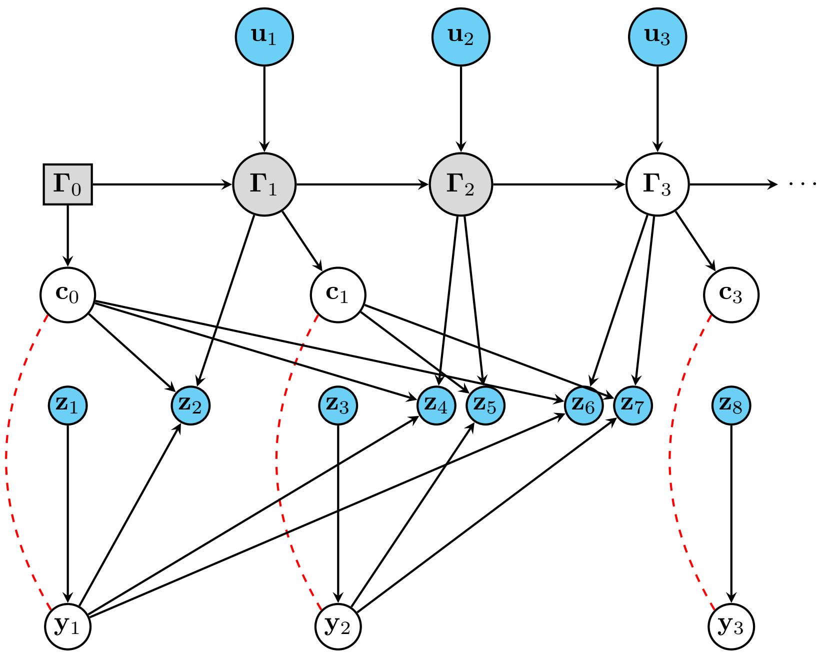


Figure 8.1: *The DBN of a small SLAM problem addressed with a framed parameterization. Gray nodes represent the robot poses that are progressively marginalized out by the prediction step (i.e., the new robot pose is predicted and the old one is discarded). A dashed red link connects each landmark, expressed with the FHP parameterization, with its anchor frame.*

poses are not updated by successive measurements, thus theirs estimates may result more imprecise than the ones obtainable if the robot poses are maintained in the filter state, alike what happens in a Kalman smoother. The use of anchored parameterization in the EKF SLAM mechanism overcomes this issue, since some past pose remains in the state vector. In the next we show how it is possible to construct an optimization graph starting from the state vector of an EKF SLAM estimation (and from its equivalent Dynamic Bayesian Network).

8.1.1 DBN analysis

Considering the so called *framed parameterizations* presented in the previous chapters (i.e., the FHP and FIS parameterizations, both in their direct and indirect formulations), where the anchor frame is shared between all the features initialized at the same time, we can observe that the DBN of the SLAM assume a particular form, since some past robot poses are maintained in the filter as *anchor frames*. This happens when new landmarks are added to the filter state: a unique new common anchor frame, which represents the current pose of the camera is added to the filter, then the landmarks are referred to this anchor frame.

Lets comment the DBN of Figure 8.1 that shows a small example of a monocular SLAM problem addressed with a framed parameterization; in the next we will use this example to introduce the proposal of the *CIBA-SLAM* method. We follow the evolution of the net during the estimation problem.

At first time step, the robot pose $\mathbf{\Gamma}_0$ is initialized and a new landmark is perceived. A new landmark ($\mathbf{y}_1$) is added to the net using measure $\mathbf{z}_1$. The use of framed parameterization allows the description of the landmark in the camera reference system, i.e., without any dependence on the current robot pose, thus the node $\mathbf{y}_1$ results connected only to the measure $\mathbf{z}_1$. This would not have happened if a point anchored parameterization (i.e., UID or AHP parameterization) had been used: the landmark description would have been built by the current robot attitude and the external measure. The camera pose $\mathbf{c}_0$ is added to the net (and to the filter) as an anchor frame thanks to the composition of current robot pose with the known transformation $\mathbf{T}^{\mathcal{R}}_{\mathcal{C}}$. Notice that here we prefer to change a little bit the notation with respect to the EKF formulation, where the anchor frames are named $\mathbf{y}^{AF}_i$ and the landmarks $\mathbf{y}^{FHP}_j$, to ease the remaining part of the chapter. The $\mathbf{T}^{\mathcal{R}}_{\mathcal{C}}$ is a known parameter and it is not drawn in the figure to avoid the use of too many nodes and arcs. A dashed red edge connects $\mathbf{y}_1$ and $\mathbf{c}_0$ to recall the relation between the landmark and its anchor frame, although it does not have any meaning in the DBN formulation.

Successively, odometric data $\mathbf{u}_1$ composed to with robot position $\mathbf{\Gamma}_0$ generate robot pose $\mathbf{\Gamma}_1$. The original pose $\mathbf{\Gamma}_0$ is marginalized out, i.e., it is no more available in the filter state, due to state completeness assumption. We enhance this fact in the DBN by filling in gray the nodes that are no more in the filter state. At this time the landmark $\mathbf{y}_1$ is observed through measure $\mathbf{z}_2$, involving the current robot pose $\mathbf{\Gamma}_1$ and the anchor frame $\mathbf{c}_0$. It has to be noticed that the transformation between the robot and the camera is needed to compute the projection of the landmark in the image, but this detail is skipped to avoid confusion in the graph. At the same time a new landmark $\mathbf{y}_2$ is initialized through measure $\mathbf{z}_3$ and the current camera pose $\mathbf{c}_1$ is generated from $\mathbf{\Gamma}_1$. In the next iteration of the SLAM algorithm, landmarks $\mathbf{y}_1$ and $\mathbf{y}_2$ are measured respectively through $\mathbf{z}_4$ and $\mathbf{z}_5$. No landmark is added and consequently no anchor frame is generated in the filter.

At last time step a new landmark $\mathbf{y}_3$ is added to the net, the camera pose $\mathbf{c}_3$ is generated as the anchor frame of the new landmark and the landmarks $\mathbf{y}_1$ and $\mathbf{y}_2$ are measured respectively through $\mathbf{z}_6$ and $\mathbf{z}_7$ in the current robot pose $\mathbf{\Gamma}_3$. At the end of the execution we have in the filter (i.e., in the final DBN) the last robot pose $\mathbf{\Gamma}_3$, the three anchor frames ($\mathbf{c}_0$,$\mathbf{c}_1$,$\mathbf{c}_3$) and the three landmarks ($\mathbf{y}_1$,$\mathbf{y}_2$,$\mathbf{y}_3$), while past robot pose nodes ($\mathbf{\Gamma}_0$, $\mathbf{\Gamma}_1$, $\mathbf{\Gamma}_2$) have been marginalized out. The state elements have been estimated with measures $\mathbf{z}_{1:8}$ and inputs $\mathbf{u}_{1:3}$, thus, for instance, the anchor frame elements, i.e., the camera poses at the times in which a landmark was firstly observed, differs at the end of the estimation process from the original pose in which they have been generated and they represent a sort of *milestones* or *keyframes* of the trajectory followed by the cameras. Notice that the granularity of the milestones depends on the frequency of landmark addition.

Let us do some considerations on the elements that are in the filter at the end, i.e., the elements that have not been marginalized out. Consider any two camera poses (i.e., two anchor frames) $\mathbf{c}_i$ and $\mathbf{c}_j$ with $j > i$. They represent the pose of the camera w.r.t. the world (or the current map) reference system, i.e., respectively the transformations $\mathbf{T}^{\mathcal{W}}_{\mathcal{C}_i}$ and $\mathbf{T}^{\mathcal{W}}_{\mathcal{C}_j}$. Moreover, they have been generated from the robot pose w.r.t. world at time i and j composed with the known transformation $\mathbf{T}^{\mathcal{R}}_{\mathcal{C}}$ and they are linked by the set of odometric

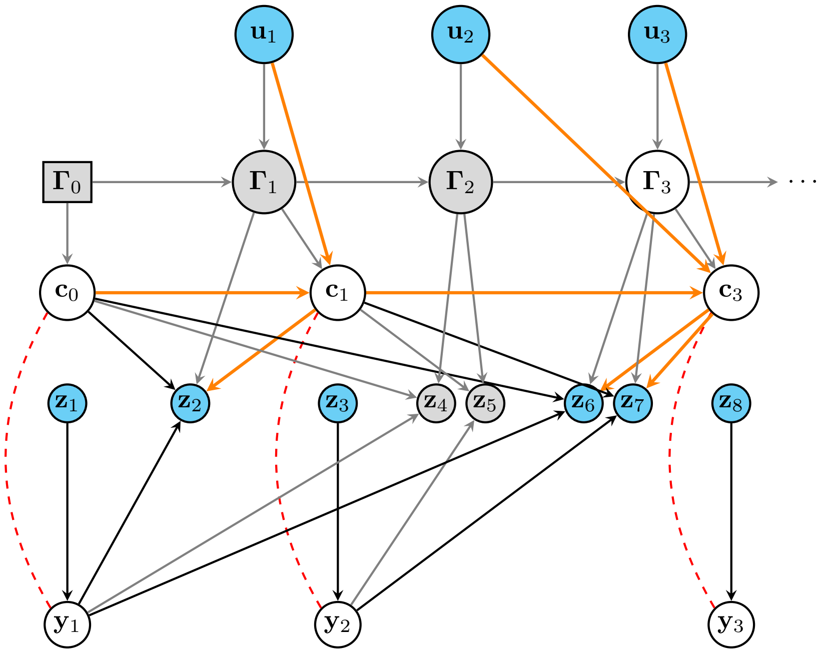


Figure 8.2: *The same DBN of Figure 8.1. Gray nodes represents the robot poses that are progressively marginalized out by the prediction step. All arrows exiting from the marginalized nodes are grayed too. A dashed red link connects each landmark, expressed with the FHP parameterization, with its anchor frame. Orange arrows highlight the hidden relations between nodes: the measurement $\mathbf{z}_i$ can be performed in the an anchor frame instead of the robot pose and the odometry can be transformed to connect camera poses instead of robot poses. Measurement $\mathbf{z}_4$ and $\mathbf{z}_5$ are grayed because they are referred to a camera pose that is not present in the filter.*

inputs $\{\mathbf{u}_{i+1:j}\}$, equivalent to the relative transformation $\mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_j}$ between the robot pose at time i and j. Consequently we can express the relation betwee the two camera poses directly in terms of the odometric inputs as

$$\mathbf{T}^{\mathcal{W}}_{\mathcal{C}_j} = \mathbf{T}^{\mathcal{W}}_{\mathcal{C}_i} \otimes \left(\mathbf{T}^{\mathcal{R}}_{\mathcal{C}}\right)^{-1} \otimes \mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_j} \otimes \mathbf{T}^{\mathcal{R}}_{\mathcal{C}}. \tag{8.1}$$

This is highlighted in Figure 8.2, that represent exactly the same net of Figure 8.1 with the addition of orange arrows linking successive camera poses and the odometric measurements to the camera poses that they generates. Notice that in this figure all the edges exiting from nodes that are not in the filter at the end of the estimation are grayed.

In the case in which odometric inputs are not used in the SLAM formulation, the input $\mathbf{u}_i$ disappear from the net, and the nodes $\mathbf{\Lambda}_i$, representing the parameters of the motion, are added. As it happens for the robot poses, only the last node of motion parameters remains in the net, while previous are marginalized.

Consider now the measurements of landmarks, and focus on, for instance, the observation $\mathbf{z}_2$. By definition, a measurement depends on the anchor frame ($\mathbf{c}_0$), on the landmark

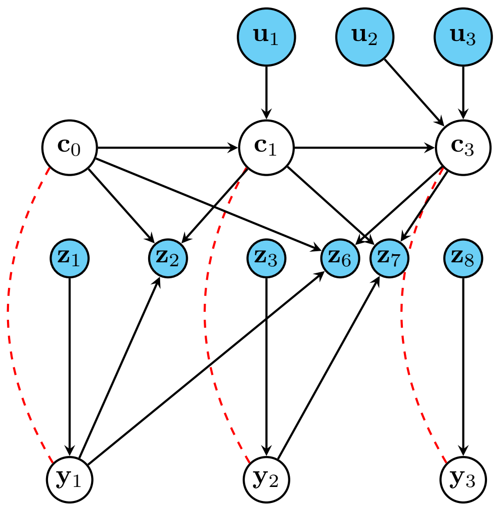


Figure 8.3: *The same net of Figure 8.2 but the marginalized nodes and the inactive measurements.*

($\mathbf{y}_1$), on the current robot pose ($\mathbf{\Gamma}_1$) and on the transformation $\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}$. Practically, the measurement is taken from the camera position, thus without considering the robot pose $\mathbf{\Gamma}_1$, we can introduce an equivalent measurement equation that measures landmark $\mathbf{y}_1$ in the camera position $\mathbf{c}_1$. The latter is generated by the composition of the robot pose $\mathbf{\Gamma}_1$ and the known transformation $\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}$. We highlight this by the addition of orange arrows to the DBN of figure Figure 8.2. This allows to express measurements in terms of nodes that are still in the DBN at the end of the estimation. Only measurements evaluated when at least a landmark has been added, i.e., on the *milestones* of the camera trajectory can be expressed in this venue, while others can not be directly expressed with the elements that are in the filter state at the end of the on-line estimation, thus their nodes are filled in gray in the DBN (look at nodes $\mathbf{z}_4$ and $\mathbf{z}_5$). We call these measurements *inactive*, since theirs values can not be recomputed with the elements that are maintained in the state vector.

8.1.2 Map Bundle Adjustment

The deletion of marginalized nodes, inactive measurements and last robot pose from the DBN of Figure 8.2 gives the net of Figure 8.3. From the on-line estimation performed with the EKF SLAM algorithm we have an estimate of the probability

$$p(\mathbf{c}_{0,1,3}, \mathbf{y}_{1,2,3} | \mathbf{z}_{1,2,3,4,5,6,7,8}, \mathbf{u}_{1,2,3}), \tag{8.2}$$

(where the last robot pose was marginalized out), while the new net describes the probability

$$p(\mathbf{c}_{0,1,3}, \mathbf{y}_{1,2,3} | \mathbf{z}_{1,2,3,6,7,8}, \mathbf{u}_{1,2,3}). \tag{8.3}$$

The two problems are not equivalent, but lets hypothesize that the dropped details (i.e., the inactive measurements $\mathbf{z}_4$ and $\mathbf{z}_5$) are not so much relevant with respect to the errors introduced by the linearizations of the EKF mechanism. Under this hypothesis we can formulate a proposal for a refinement of the estimation as it follows.

We can define a graph for a Non Linear Least Square optimization which mimics the structure of the considered DBN, where nodes are camera poses (i.e., anchor frames) and landmarks and edges represent measurements and odometric information. The initialization for the optimization process is directly taken from the mean values estimated in the EKF machinery, while, after the optimization, the resulting estimates are reinserted in the EKF, both in terms of mean values and covariances. In the next this method will be detailed properly, starting from the precise formulation of the structure of the graph to optimize.

Starting from the DBN presented above, it is possible to create an equivalent graph by considering all the camera nodes $\mathbf{c}_i$ and the the landmark nodes $\mathbf{y}_j$ in the DBN and by connecting them with edges representing the measurements $\mathbf{z}$. In particular, we have two different kind of edges. An unary edge $\mathbf{e}^I_{\mathbf{y}_i}$ expresses the first measurement of the landmark, i.e., its position in the first image where it has been perceived; this is independent from the camera position. A ternary edge $\mathbf{e}^M_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}$ represents the measure of the landmark $\mathbf{y}_j$, anchored to the camera position $\mathbf{c}_i$, in the image of the camera $\mathbf{c}_k$. Odometric inputs are accumulated when camera nodes are not consecutive, expressed into the camera reference system and expressed in the form of binary edges $\mathbf{e}^O_{\mathbf{c}_i\mathbf{c}_j}$ (e.g., the edge $\mathbf{e}^O_{\mathbf{c}_1\mathbf{c}_3}$ is generated by $\mathbf{u}_2$ and $\mathbf{u}_3$). Remember that odometry edges may not be present in the graph depending on the specific problem.

The result of the conversion of the DBN represented in Figure 8.3 in a corresponding optimization graph is shown in Figure 8.4. It has to be noticed that this is a particular formulation of the Bundle Adjustment problem. Standard approaches consider landmarks as 3D points connected to cameras by binary edges representing measurements, as in [95]. With standard formulation it is intuitively more difficult to propagate the result of the Bundle Adjustment to the EKF state, which is the goal of our approach. Indeed, the proposed formulation create a graph that reflects directly the DBN structure and this allows to treat easily the multi camera SLAM setup too. Assuming that the landmarks are initialized in a single camera, observations in different cameras are representable in the DBN with additional measurement nodes that can be easily translated in edges in the graph formulation.

In the remaining of this chapter we describe all the details of our approach: firstly, the mathematical formulation of the nodes and the edges will be given, then the integration in the CI-SLAM framework is presented.

8.2 Graph Formalization

We want to maintain a structure of the nodes in the graph that is more similar as possible to the EKF state description. We consider the *Hybrid Indirect EKF-SLAM* formulation presented in Chapter 7 and in particular the FHP$_i$ parameterization, later we extend the formulation to the use of FIS$_i$ parameterization. The direct formulation of the EKF-SLAM is discarded since it is known to have problems in managing rotations due to the presence of quaternions in the CI SLAM setup.

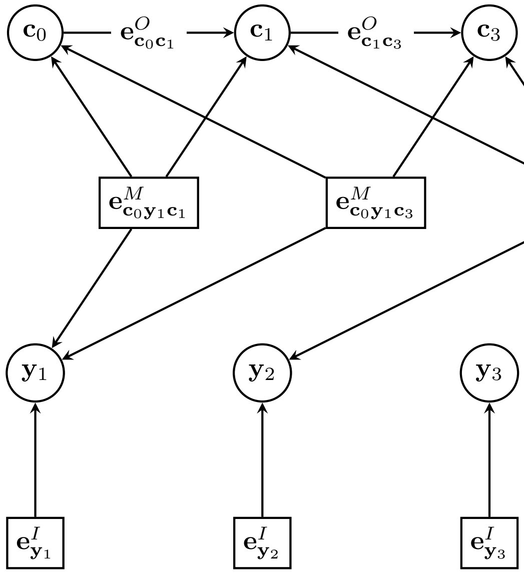

$\mathbf{c}_0$
$\mathbf{e}^O_{\mathbf{c}_0\mathbf{c}_1}$
$\mathbf{c}_1$
$\mathbf{e}^O_{\mathbf{c}_1\mathbf{c}_3}$
$\mathbf{c}_3$
$\mathbf{e}^M_{\mathbf{c}_0\mathbf{y}_1\mathbf{c}_1}$
$\mathbf{e}^M_{\mathbf{c}_0\mathbf{y}_1\mathbf{c}_3}$
$\mathbf{e}^M_{\mathbf{c}_1\mathbf{y}_2\mathbf{c}_3}$
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_3$
$\mathbf{e}^I_{\mathbf{y}_1}$
$\mathbf{e}^I_{\mathbf{y}_2}$
$\mathbf{e}^I_{\mathbf{y}_3}$

Figure 8.4: *The graph corresponding to DBN of Figure 8.3 where only the elements available in the state vector after the estimation are used, resulting in a sub problem of the full SLAM graph since camera node $\mathbf{c}_2$ and relative measurements are discarded.*

8.2.1 Nodes in the graph

Recalling the indirect formulation, the variables in the EKF state vector that have to be converted to nodes are the set of anchor frames $\tilde{\mathbf{y}}_i^{AF_i}$ and the set of FHP$_i$ landmarks $\tilde{\mathbf{y}}_j^{FHP_i}$. Each of them can be decomposed in the nominal part and in the error state part respectively as $\tilde{\mathbf{y}}_i^{AF_i} = \mathbf{y}_i^{AF_i} \boxplus \delta\mathbf{y}_i^{AF_i}$ and $\tilde{\mathbf{y}}_j^{FHP_i} = \mathbf{y}_j^{FHP_i} \boxplus \delta\mathbf{y}_j^{FHP_i}$. To comply with the Hybrid Indirect EKF-SLAM formulation we can not change the values of the nominal part of the landmarks, i.e., $\mathbf{y}_i^{AF_i}$ and $\mathbf{y}_j^{FHP_i}$. Consequently we want that only the error state part will be updated during the iteration of the optimization algorithm.

Considering a generic node $\mathbf{x}$ in the graph, we have to consider that it is composed by three elements: $\mathbf{x}^N$ is the nominal values that will never change, $\mathbf{x}^\delta$ is the part subject to the optimization and $\Delta\mathbf{x}$ are the increment computed in the optimization process and integrated in $\mathbf{x}^\delta$ at each iteration. Thus, we redefine the composition operator as $\overset{\triangle}{\boxplus}$ to update the error state part with a NNLS techniques as $\mathbf{x}^{\delta^+} = \mathbf{x}^\delta \overset{\triangle}{\boxplus} \Delta\mathbf{x}$, while the complete composition is given by $\mathbf{x}^N \boxplus \mathbf{x}^{\delta^+} = \mathbf{x}^N \boxplus \left(\mathbf{x}^\delta \overset{\triangle}{\boxplus} \Delta\mathbf{x}\right)$.

8.2.1.1 The camera pose node

The camera pose node $\mathbf{c}_i$, equivalent to an anchor frame in the EKF, is composed by three different parts as

$$\mathbf{c}_i = \mathbf{c}_i^N \boxplus \mathbf{c}_i^\delta \overset{\triangle}{\boxplus} \Delta\mathbf{c}_i, \tag{8.4}$$

and we can further split it in the translation and rotation terms

$$\mathbf{t}_i = \mathbf{t}_i^N + \mathbf{t}_i^\delta + \Delta\mathbf{t}_i, \tag{8.5}$$

$$\mathbf{q}_i = \mathbf{q}_i^N \otimes \begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_i^\delta}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\boldsymbol{\theta}_i}{2} \end{bmatrix}. \tag{8.6}$$

The $\mathbf{c}_i^N$ represents the nominal values and it is copied from the values of the $\mathbf{y}_i^{AF_i}$, i.e., the nominal values of the EKF state variable. It is represented by a translation vector $\mathbf{t}_i^N$ and a quaternion $\mathbf{q}_i^N$; these elements will be never modified by the optimization, i.e., they act as a sort of fixed parameters. The $\mathbf{c}_i^\delta$ element corresponds to the error state part $\delta\mathbf{y}^{AF_i}$ of the anchor frame in the EKF state. It is composed by a translation vector $\mathbf{t}_i^\delta$ and a rotation vector $\boldsymbol{\theta}_i^\delta$. Finally the $\Delta\mathbf{c}_i$ part is composed by a translation vector $\Delta\mathbf{t}_i$ and a rotation vector $\Delta\boldsymbol{\theta}_i$.

At each iteration of the optimization algorithm the element $\mathbf{c}_i^\delta$ has to be composed with the increment $\Delta\mathbf{c}_i$, thus we have to define the $\overset{\Delta}{\boxplus}$ operator. The composition of the translation part is simply given by the vector sum

$$\mathbf{t}_i^{\delta^+} = \mathbf{t}_i^\delta + \Delta\mathbf{t}_i. \tag{8.7}$$

The composition of the local rotational part can be performed by considering local approximation of the quaternions

$$\begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_i^{\delta^+}}{2} \end{bmatrix} = \eta \begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_i^\delta}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \Delta\frac{\boldsymbol{\theta}_i}{2} \end{bmatrix}, \tag{8.8}$$

resulting in

$$\eta = \frac{1}{1 + \boldsymbol{\theta}_i^{\delta^T} \Delta\boldsymbol{\theta}_i}, \tag{8.9}$$

$$\boldsymbol{\theta}_i^{\delta^+} = \eta \left(\Delta\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^\delta + \begin{bmatrix} \frac{\boldsymbol{\theta}_i^\delta}{2} \end{bmatrix}_\times \Delta\boldsymbol{\theta} \right). \tag{8.10}$$

The complete composition is given by $\mathbf{c}_i^{\delta^+} = \mathbf{c}_i^\delta \overset{\Delta}{\boxplus} \Delta\mathbf{c}_i = \begin{bmatrix} \mathbf{t}_i^{\delta^+} \\ \frac{\boldsymbol{\theta}_i^{\delta^+}}{2} \end{bmatrix}$.

8.2.1.2 The landmark node

A landmark is represented by nodes $\mathbf{y}_i$ that corresponds to the indirect FHP$_i$ parameterization. The composition of nominal values, error state element and increments is simply given by the sum:

$$\mathbf{y}_i = \mathbf{y}_i^N + \mathbf{y}_i^\delta + \Delta\mathbf{y}. \tag{8.11}$$

All the involved terms have three elements, two representing the coordinate on the normalized image plane of the viewing ray of the landmark and the third representing the inverse

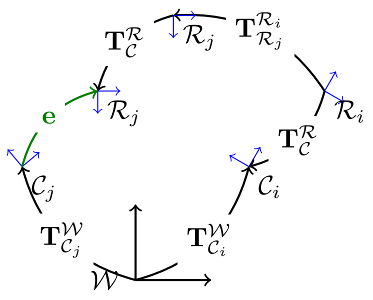

$\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}$
$\mathcal{R}_j$
$\mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i}$
$\mathbf{e}$
$\mathcal{R}_j$
$\mathcal{R}_i$
$\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}$
$\mathcal{C}_j$
$\mathcal{C}_i$
$\mathbf{T}_{\mathcal{C}_j}^{\mathcal{W}}$
$\mathbf{T}_{\mathcal{C}_i}^{\mathcal{W}}$
$\mathcal{W}$

Figure 8.5: *Frame transformations that express the odometric edge formulation.*

distance: the nominal vector $\mathbf{y}_i^N$ is composed by $\boldsymbol{p}_{\pi_i}^N$ and w_i^N, the error state vector $\mathbf{y}_i^\delta$ is composed by $\boldsymbol{p}_\pi^\delta$ and w_i^δ and the increments $\Delta\mathbf{y}_i$ are composed by $\Delta\boldsymbol{p}_{\pi_i}$ and Δw_i. The composition of the increments with the current value is simply given by

$$\mathbf{y}_i^{\delta^+} = \mathbf{y}_i^\delta + \Delta\mathbf{y}. \tag{8.12}$$

8.2.2 The Odometric Edge

As already stated, we have three different types of edges: one represents the odometric measure, one represents the measurement at the landmark initialization, i.e., its perception in the first frame, the last represents the observation of landmarks in a different camera pose. Here we give the complete mathematical formulation of the edges and we introduce a fourth edge that represents the measurement of landmarks in a multi camera setup.

Odometric measurements are translated in binary edges $\mathbf{e}_{\mathbf{c}_i\mathbf{c}_j}^O$,with $j > i$, that links two camera poses. The edge formulation has to express the error $\mathbf{e}$ between relative camera poses, expressed by the two nodes $\mathbf{c}_i$ and $\mathbf{c}_j$, and the relative transformation expressed by the odometry. By manipulating Equation 8.1 we obtain

$$\mathbf{e} = \left(\mathbf{T}_{\mathcal{C}_j}^{\mathcal{W}}\right)^{-1} \otimes \mathbf{T}_{\mathcal{C}_i}^{\mathcal{W}} \otimes \left(\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}\right)^{-1} \otimes \mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i} \otimes \mathbf{T}_{\mathcal{C}}^{\mathcal{R}}. \tag{8.13}$$

This can be verified in Figure 8.5, where the transformation composition can be followed graphically. Notice that we have transformed the odometry between two robot poses $\mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i}$ in camera coordinate system by the known transformation $\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}$. The error is null (i.e., it represents the identity transformation) when the camera nodes satisfy the measurements provided by the odometry.

Lets details better the error formulation. First of all we can express the odometry in the camera reference frame as the transformation

$$\mathbf{T}_{odo} \quad = \quad \left(\mathbf{T}_{\mathcal{C}}^{\mathcal{R}}\right)^{-1} \otimes \mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i} \otimes \mathbf{T}_{\mathcal{C}}^{\mathcal{R}}, \tag{8.14}$$

that expresses the relative position of camera $\mathbf{c}_j$ w.r.t $\mathbf{c}_i$ with the odometric information. The transformation composition of Equation 8.13, expressed with homogeneous matrices results

$$\begin{bmatrix} \mathbf{R}_{\mathcal{C}_j}^{\mathcal{W}^T} & -\mathbf{R}_{\mathcal{C}_j}^{\mathcal{W}^T}\mathbf{t}_{\mathcal{C}_j}^{\mathcal{W}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{C}_i}^{\mathcal{W}} & \mathbf{t}_{\mathcal{C}_i}^{\mathcal{W}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{odo} & \mathbf{t}_{odo} \\ 0 & 1 \end{bmatrix}. \tag{8.15}$$

Considering only the rotation part we obtain

$$\mathbf{R}_{\mathbf{e}} \quad = \quad \mathbf{R}_{\mathcal{C}_j}^{\mathcal{W}^T} \mathbf{R}_{\mathcal{C}_i}^{\mathcal{W}} \mathbf{R}_{odo}, \tag{8.16}$$

and coming back to quaternions gives

$$\mathbf{q}_{\mathbf{e}} \quad = \quad \mathbf{q}_{\mathcal{C}_j}^{\mathcal{W}^*} \otimes \mathbf{q}_{\mathcal{C}_i}^{\mathcal{W}} \otimes \mathbf{q}_{odo} \;=\; \left[\mathbf{q}_w \, \mathbf{q}_{\boldsymbol{\theta}}^T\right]^T. \tag{8.17}$$

When the relative orientation of the two camera nodes satisfies the relative orientation measured by the odometric information, the previous formula results in the identity. Thus we can express the error in the rotation with a rotation vector $\boldsymbol{\theta}_{\mathbf{e}}$, leveraging on the small rotation approximations equations, as

$$\begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_{\mathbf{e}}}{2} \end{bmatrix} = \eta \, \mathbf{q}_{\mathbf{e}} \quad = \quad \eta \, \mathbf{q}_{\mathcal{C}_j}^{\mathcal{W}^*} \otimes \mathbf{q}_{\mathcal{C}_i}^{\mathcal{W}} \otimes \mathbf{q}_{odo}, \tag{8.18}$$

i.e., the error is

$$\boldsymbol{\theta}_{\mathbf{e}} \quad = \quad 2 \, \eta \, \mathbf{q}_{\boldsymbol{\theta}}, \tag{8.19}$$

$$\tag{8.20}$$

with

$$\eta \quad = \quad \frac{1}{\mathbf{q}_w}. \tag{8.21}$$

For the translation part we obtain from Equation 8.15 the error as

$$\mathbf{e}_{\mathbf{t}} \quad = \quad \mathbf{R}_{\mathcal{C}_j}^{\mathcal{W}^T} \mathbf{R}_{\mathcal{C}_i}^{\mathcal{W}} \mathbf{t}_{odo} + \mathbf{R}_{\mathcal{C}_j}^{\mathcal{W}^T} \left(\mathbf{t}_{\mathcal{C}_i}^{\mathcal{W}} - \mathbf{t}_{\mathcal{C}_j}^{\mathcal{W}} \right) \tag{8.22}$$

The complete error function of the edge is composed by the 6-element vector

$$\mathbf{e}_{\mathbf{c}_i \mathbf{c}_j}^{O} \quad = \quad e_{odo}\left(\mathbf{c}_i, \mathbf{c}_j, \mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i}\right) \quad = \quad \begin{bmatrix} \mathbf{e}_{\mathbf{t}} \\ \boldsymbol{\theta}_{\mathbf{e}} \end{bmatrix} \tag{8.23}$$

but we have to consider the following substitution that comes from the camera nodes formulation introduced in Section 8.2.1.1:

$$\mathbf{T}_{\mathcal{C}_i}^{\mathcal{W}} \quad = \quad \mathbf{c}_i^N \boxplus \mathbf{c}_i^\delta \overset{\triangle}{\boxplus} \Delta \mathbf{c}_i \tag{8.24}$$

$$\mathbf{T}_{\mathcal{C}_j}^{\mathcal{W}} \quad = \quad \mathbf{c}_j^N \boxplus \mathbf{c}_j^\delta \overset{\triangle}{\boxplus} \Delta \mathbf{c}_j \tag{8.25}$$

We recall that to perform the optimization the Jacobians of the edge error function with respect to $\Delta\mathbf{c}_i$ and $\Delta\mathbf{c}_j$ has to be computed and evaluated in the nominal values $\mathbf{c}_i^N$, $\mathbf{c}_j^N$, with the current estimation $\mathbf{c}_i^\delta$, $\mathbf{c}_j^\delta$ and considering null increments.

8.2.2.1 Odometry composition

The edge that express the relative position of two cameras may involve more than one odometric measurement. This because cameras are milestones in the trajectory, taken when at least a new landmark has been added to the filter. In addition to the transformation in the camera reference frame, odometric data have to be iteratively composed and the covariances properly propagated. The odometric inputs $\mathbf{u}_{i+1,\dots,j}$, split in the translation part $\Delta\mathbf{t}_{i+1,\dots,j}$ and rotation part $\Delta\mathbf{r}_{i+1,\dots,j}$ can be iteratively composed in the transformation $\mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_k}$, in turn composed by the translation vector $\mathbf{t}^{\mathcal{R}_i}_{\mathcal{R}_k}$ and the rotation vector $\boldsymbol{\theta}^{\mathcal{R}_i}_{\mathcal{R}_k}$ as

$$\mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_k} = \begin{bmatrix} \mathbf{t}^{\mathcal{R}_i}_{\mathcal{R}_k} \\ \boldsymbol{\theta}^{\mathcal{R}_i}_{\mathcal{R}_k} \end{bmatrix} = c_{odo}\left(\mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_{k-1}}, \mathbf{u}_k\right) = \begin{cases} \mathbf{t}^{\mathcal{R}_i}_{\mathcal{R}_{k-1}} + \mathbf{R}\left(\mathbf{q}\right)\Delta\mathbf{t}_k \\ 2\eta\,\mathbf{q}_{\boldsymbol{\theta}} \end{cases} \tag{8.26}$$

where

$$\eta = \frac{1}{\mathbf{q}_w} \tag{8.27}$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_w \\ \mathbf{q}_{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_{odo}}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\mathbf{r}_k}{2} \end{bmatrix}. \tag{8.28}$$

The composition has to be iterated on $k = i+1,\dots,j$ starting with the identity transformation $\mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_i}$, i.e., a null vector for $\mathbf{t}^{\mathcal{R}_i}_{\mathcal{R}_i}$ and $\boldsymbol{\theta}^{\mathcal{R}_i}_{\mathcal{R}_i}$. To compose the covariance of the odometry it is necessary to evaluate the Jacobians of the previous function and to compose iteratively a covariance matrix. In particular, starting with a null 6×6 covariance matrix $\boldsymbol{\Sigma}^{\mathcal{R}_i}_{\mathcal{R}_i}$, we have to iterate

$$\boldsymbol{\Sigma}^{\mathcal{R}_i}_{\mathcal{R}_k} = \frac{\partial c_{odo}(\cdots)}{\partial \mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_{k-1}}}\boldsymbol{\Sigma}^{\mathcal{R}_i}_{\mathcal{R}_{k-1}}\frac{\partial c_{odo}(\cdots)}{\partial \mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_{k-1}}}^T + \frac{\partial c_{odo}(\cdots)}{\partial \mathbf{u}_k}\boldsymbol{\Sigma}_k\frac{\partial c_{odo}(\cdots)}{\partial \mathbf{u}_k}^T \tag{8.29}$$

for $k = i+1,\dots,j$, where $\boldsymbol{\Sigma}_k$ are the covariance matrices of the odometric inputs and the Jacobians are evaluated at current odometric input and cumulated odometry values. The composition results has to be transformed in camera coordinate reference frame with Equation 8.14.

8.2.2.2 Edge Information Matrix

The Equation 8.23 defines the edge error function, involving two camera nodes and the (cumulated) odometry between them, but we need to specify the information matrix $\boldsymbol{\Omega}^{O}_{\mathbf{c}_i\mathbf{c}_j}$ associated with the edge, i.e., the inverse of the covariance matrix $\boldsymbol{\Sigma}^{O}_{\mathbf{c}_i\mathbf{c}_j}$. The covariance matrix $\boldsymbol{\Sigma}^{O}_{\mathbf{c}_i\mathbf{c}_j}$ can be expressed through the linearization of the error function w.r.t. the odometric data, since it has an associated covariance $\boldsymbol{\Sigma}^{\mathcal{R}_i}_{\mathcal{R}_k}$. The information matrix can be computed as

$$\boldsymbol{\Omega}^{O}_{\mathbf{c}_i\mathbf{c}_j} = \left(\frac{\partial e_{odo}\left(\cdots\right)}{\partial \mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_j}}\boldsymbol{\Sigma}^{\mathcal{R}_i}_{\mathcal{R}_j}\frac{\partial e_{odo}\left(\cdots\right)}{\partial \mathbf{T}^{\mathcal{R}_i}_{\mathcal{R}_j}}^T\right)^{-1} \tag{8.30}$$

Notice that at each iteration of the optimization algorithm the information matrix has to be updated, since the changes in the camera nodes $\mathbf{c}_i$ and $\mathbf{c}_j$ affect the covariance propagation of Equation 8.30.

8.2.3 Landmark Initialization Edge

The first type of edge that involves landmarks is the initialization edge. A landmark $\mathbf{y}_i = [\mathbf{p}_{\pi_i}^T,\, w_i]$ is expressed in the camera reference frame as the position of the image point on the normalized image plane ($\mathbf{p}_{\pi_i}$) (i.e., the first two elements of the direction vector) with its inverse scale factor (w_i). The unary edge on the landmark acts as a sort of constraint on the direction vector of the landmark, specified at the first landmark perception. Since at initialization the depth is not observable, this edge does not put any constraint on it. The error function of the edge is

$$\mathbf{e}_{\mathbf{y}_i}^I \quad = \quad K(\mathbf{p}_{\pi_i}) - \boldsymbol{p}_{img} + \boldsymbol{\delta}, \tag{8.31}$$

where $\boldsymbol{p}_{img}$ is the image point in which the landmark was firstly perceived (i.e., the salient point detector output) and $\boldsymbol{\delta}$ is a Gaussian zero mean bivariate noise with $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$ that represents the uncertainty on the perception. We remind that the landmark $\mathbf{y}_i$ is composed by $\mathbf{y}_i^N + \mathbf{y}_i^\delta + \Delta\mathbf{y}_i$, thus

$$\mathbf{p}_{\pi_i} \quad = \quad \mathbf{p}_{\pi_i}^N + \mathbf{p}_{\pi_i}^\delta + \Delta\mathbf{p}_{\pi_i}. \tag{8.32}$$

The evaluation of the Jacobian of the error function has to be performed with respect to the $\Delta\mathbf{y}_i$ element, resulting in a 2×3 matrix where the last column is zero, since the inverse scale element is not involved in the edge error function. The information matrix of the edge is simply given by the inversion of the covariance matrix of the $\boldsymbol{\delta}$ noise:

$$\boldsymbol{\Omega}_{\mathbf{y}_i}^I \quad = \quad \boldsymbol{\Sigma}_{\boldsymbol{\delta}}^{-1}. \tag{8.33}$$

A common choice is to consider $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$ as a identity matrix, representing a 1 pixel standard deviation.

8.2.4 Landmark Measurement Edge

A landmark measurement edge aims at the evaluation of the projection error of the landmark $\mathbf{y}_j$, referred to the camera pose $\mathbf{c}_i$, in the camera at pose $\mathbf{c}_k$. The equation of the error, which results very similar to the measurement equation of an FHP$_i$ landmark in the EKF formulation, is

$$\mathbf{e}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}^M \quad = \quad K\left(\mathbf{R}\left(\mathbf{q}_k\right)^T\left(w_j\left[\mathbf{t}_i - \mathbf{t}_k\right] + \mathbf{R}\left(\mathbf{q}_i\right)\begin{bmatrix}\boldsymbol{p}_{\pi_j}\\1\end{bmatrix}\right)\right) - \mathbf{z}_{jk}^I + \boldsymbol{\delta}, \tag{8.34}$$

where it has to be remembered that $\mathbf{c}_i$ can be decomposed in the translation vector $\mathbf{t}_i$ and in the quaternion $\mathbf{q}_i$ resulting from the composition $\mathbf{c}_i^N \boxplus \mathbf{c}_i^\delta \overset{\Delta}{\boxplus} \Delta\mathbf{c}_i$ (see Section 8.2.1.1). The same consideration has to be applied to the second camera $\mathbf{c}_k$, while, for the landmark $\mathbf{y}_j$ the formulation is simpler, since only a summation of the nominal, delta and increment terms is requested. To perform optimization, the Jacobians of $\mathbf{e}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}^M(\cdots)$ w.r.t. the two cameras increments $\Delta\mathbf{c}_i$ and $\Delta\mathbf{c}_k$ and the landmark $\Delta\mathbf{y}_j$ increment have to be computed.

The terms $\boldsymbol{\delta}$ is a zero mean bivariate Gaussian noise with covariance $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$ that represents the error on the measurement process. Since it is additive the information matrix associated with the edge is simply the inverse of the covariance matrix. Usually it is considered an identity matrix, representing a 1 pixel standard deviation in the measurement process.

8.2.5 Landmark Measurement In Multi Camera Setup

The landmark measurement edge presented in the previous section assumes that the camera pose in which the landmark was initialized and the camera pose in which it is measured are the same physical camera, i.e., they have the same calibration. In a multi camera setup a landmark $\mathbf{y}_j$, initialized in camera pose $\mathbf{c}_i$, may be observed by a different camera, which pose is given by the composition of the camera pose $\mathbf{c}_k$ and the relative transformation $\mathbf{T}^{\mathcal{C}_I}_{\mathcal{C}_O}$ where the $\mathcal{C}_I$ indicates the camera in which the landmark has been initialized and $\mathcal{C}_O$ indicates the observing camera. The projection parameters are encoded in the $K_O(\cdots)$ function. The resulting edge equation $\mathbf{e}^{M_O}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}$ is

$$K_O\left(\mathbf{R}(\mathbf{q}^{\mathcal{C}_I}_{\mathcal{C}_O})^T\mathbf{R}\left(\mathbf{q}_k\right)^T\left(w_j\left[\mathbf{t}_i-\mathbf{t}_k\right]+\mathbf{R}\left(\mathbf{q}_i\right)\begin{bmatrix}\boldsymbol{p}_\pi\\1\end{bmatrix}\right)-w_j\mathbf{R}(\mathbf{q}^{\mathcal{C}_I}_{\mathcal{C}_O})^T\mathbf{t}^{\mathcal{C}_I}_{\mathcal{C}_O}\right)-\mathbf{z}^O_{jk}+\boldsymbol{\delta} \tag{8.35}$$

If the multi camera setup involves n different cameras, for each tern $\mathbf{c}_i, \mathbf{y}_j, \mathbf{c}_k$ it is possible to have $n-1$ edges $\mathbf{e}^{M_{O_s}}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}$, i.e., one edge for each observation performed in a different camera. This is very similar to what happens in the multi camera EKF SLAM setup, where more measurements per landmark can be performed.

The optimization requires the evaluation of the Jacobians w.r.t. the camera increments $\Delta\mathbf{c}_i$, $\Delta\mathbf{c}_k$ and landmark increments $\Delta\mathbf{y}_j$. The information matrix is simply given by the inversion of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$ associated to the $\boldsymbol{\delta}$ additive noise, that is usually chosen as a 2×2 identity matrix.

8.2.6 Graph Initialization And Optimization Strategies

In Section 8.1 we have introduced how it is possible to convert a DBN representing the final estimation of an EKF slam machinery into a graph for the optimization. Here we formalize the graph construction process through the formal definition of an algorithm.

First of all we have to add two small modifications to the EKF SLAM algorithm. As the EKF SLAM algorithm starts we add an anchor even if no landmarks has been added: we can consider this camera position as a fixed frame, since it is local and it has no uncertainty. In particular, its very uncommon that in the first frame no landmarks are initialized, but this can happen especially when the submapping with the CI-SLAM framework is considered. In this case it is possible that a sufficient number of landmarks shared with the previous map prevent new landmarks addition. The second modification is to add the last anchor frame to the filter, even if no landmarks are added. This allows to consider the last camera pose in the graph optimization, i.e., to provide a graph that covers the estimation process from the start to end, although some intermediate pose of the camera and its associated observations are dropped if compared to a full SLAM estimation problem.

With reference to Algorithm 9, let details precisely how to construct a graph from an EKF state vector, how to initialize properly the nodes values, (i.e., how to provide a valid initialization point to the NNLS algorithm), and which nodes need to be fixed in order to perform the optimization. For each anchor frame in the EKF state vector a camera node has to be inserted in the graph $\mathcal{G}$ (step 4). These nodes have to be initialized with nominal and delta values from the EKF estimate (step 6 and 7). Similarly, a FHP landmark node $\mathbf{y}_j$ has to be created for each landmark in the state vector (step 10); also in this case the initial estimation is provided by the nominal and delta values (step 12 and 13). If the

Algorithm 9 EKF2Graph

```
1: $\mathcal{G} = \emptyset,\ \mathcal{E} = \emptyset$
2: # ***Nodes Creation***
3: **for all** $\mathbf{y}^{AF}$ in EKF state vector **do**
4:     $\mathcal{G} \leftarrow \{\mathcal{G}, \text{new}\mathbf{c}_i\}$
5:     $i \leftarrow i + 1$
6:     $\mathbf{c}_i^N = \mathbf{y}^{AF}$
7:     $\mathbf{c}_i^\delta = \delta\mathbf{y}^{AF}$
8: **end for**
9: **for all** $\mathbf{y}^{FHP}$ in EKF state vector **do**
10:     $\mathcal{G} \leftarrow \{\mathcal{G}, \text{new}\mathbf{y}_j\}$
11:     $j \leftarrow j + 1$
12:     $\mathbf{y}_j^N = \mathbf{y}^{FHP}$
13:     $\mathbf{y}_j^\delta = \delta\mathbf{y}^{FHP}$
14: **end for**
15: # ***Edges addition***
16: **if** $\exists$ odometry information **then**
17:     **for all** $\mathbf{c}_i, \mathbf{c}_j, j > i \geq 0, \nexists\mathbf{c}_r, i < r < j$ **do**
18:         cumulate odometry $\mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_i}$ by $u_{i+1,\ldots,j}$
19:         $\mathcal{E} \leftarrow \left\{\mathcal{E}, \mathbf{e}^O_{\mathbf{c}_i\mathbf{c}_j}\right\}$
20:     **end for**
21: **end if**
22: **for all** $\mathbf{y}_i$ in $\mathcal{G},\ i \geq 0$ **do**
23:     $\mathcal{E} \leftarrow \left\{\mathcal{E}, \mathbf{e}^I_{\mathbf{y}_i}\right\}$
24: **end for**
25: **for all** $\mathbf{c}_i, \mathbf{y}_j, \mathbf{c}_k,\ i, j, k \geq 0$ **do**
26:     **if** $\exists$ $\mathbf{z}^I_{jk}$ in $C_I$ **then**
27:         $\mathcal{E} \leftarrow \left\{\mathcal{E}, \mathbf{e}^M_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}\right\}$
28:     **end if**
29:     **for all** cameras $C_O, C_O \neq C_I$ **do**
30:         **if** $\exists$ $\mathbf{z}^O_{jk}$ in $C_O$ **then**
31:             $\mathcal{E} \leftarrow \left\{\mathcal{E}, \mathbf{e}^{M_O}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}\right\}$
32:         **end if**
33:     **end for**
34: **end for**
35: # ***Start up***
36: Fix node $\mathbf{c}_0$ in $\mathcal{G}$
37: **if** Monocular setup without odometry **then**
38:     Fix $\mathbf{y}_k$ in $\mathcal{G}$, $k$ : $\forall h \#\,\text{meas}(\mathbf{y}_h) \leq \#\,\text{meas}(\mathbf{y}_k), h, k \geq 0$
39: **end if**
40: remove nodes with few measurements
```

data is available the pairs of successive camera poses are selected (step 17), the cumulate odometry is computed (step 18) and the edge is added to the set of edges $\mathcal{E}$ (step 19). For each landmark node an edge $\mathbf{e}^{I}_{\mathbf{y}_i}$ is added (step 23). Then for all available measurements performed in the same camera of the initialization of a landmark, an edge $\mathbf{e}^{M}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}$ is added (step 27). For measurements performed in cameras different from the one used to initialize landmarks an edge $\mathbf{e}^{M_O}_{\mathbf{c}_i\mathbf{y}_j\mathbf{c}_k}$ is added (step 31).

To perform the optimization on the graph we have to fix an adequate number of degree of freedom to prevent the singularity of the Hessian Matrix, as illustrated in Section 5.5.3. A SLAM problem has 6 free degree of freedom, since the map can be rotated and translated obtaining the same solution. Moreover, if we setup a pure Monocular SLAM problem, i.e., we have no edges on the odometry, we obtain an additional degree of freedom, that comes from the unobservable scale of the system. The first 6 degree can be fixed by specifying a camera as fixed in the graph, i.e., by excluding it from the optimization. In particular, we fix the first camera node, i.e., the position of the camera when the map start, that is fixed in the EKF too, since it has no uncertainty (step 36).

In a pure monocular setup we have to fix an additional degree of freedom. From a theoretical point of view, we can fix the depth of one landmark to determine the entire scale, but practical considerations, also related to implementation strategies, suggest to fix an entire node. We choose to fix a landmark and in particular the node that is involved in the highest number of edges, i.e., that has been measured the greatest number of times (step 38). With this choice two excessive degrees of freedom are fixed, but we consider this a good compromise: well known solution in literature (e.g., [95]) fix two consecutive cameras in the monocular contest, with a surplus of 5 degree fixes.

Step 40 aims at the deletion of the nodes that are measured an insufficient number of times. Landmarks that are only initialized, but never measured result disconnected from the rest of the optimization, thus they are removed. Similarly, landmarks that are measured a very few times (e.g., 2 or 3 times) may results in a very poor estimation, thus we consider a threshold on the number of measurement to remove nodes from the graph.

8.2.6.1 Usage with FIS parameterization

Up to here it was supposed to start from an indirect EKF SLAM setup with FHP parameterization. The graph construction can be easily adapted to the FIS parameterization. In particular, we can create exactly the same graph, but the initialization of landmark nodes $\mathbf{y}_j$ is performed by copying the inverse scale value from the FIS element in the EKF state vector and by calculating the direction vector from the initial point perception, that is stored outside the filter in the FIS parameterization. The graph optimization allows the estimation of this direction vector, which is considered exact by the FIS model and this allows to overcome a possible issue with the FIS parameterization, i.e., to correct the unmodeled error on the direction of the viewing ray.

8.2.7 Sub map setup

Lets consider to use the proposed graph construction algorithm with a state vector corresponding to a submap different from the first one. Let indicate the camera nodes and the landmarks that were initialized in a previous map respectively with $\mathbf{c}_i$, with $i < 0$, and $\mathbf{y}_j$, with $j < 0$ and let refers to them informally as "old" nodes. Intuitively there is an high probability of dropping too many measures of old landmarks in the graph formulation. For instance, consider a landmark $\mathbf{y}_i$ initialized very far in the past (e.g., a landmark on the sky in an outdoor setup); it remains visible in the camera for long time, while other landmarks disappear quickly (e.g., landmarks on walls where robot is passing by). When a new map is created the landmark $\mathbf{y}_i$ will be shared, but lots of camera poses in which it has been measured will not be shared with the new map. When this new map will be closed, the optimization graph looses a lot of information about the "old" landmarks and consequently its optimization may result in a wrong solution.

In the EKF mechanism the information about old measurements is summarized by correlations (i.e., covariances). We propose here the formulation of a particular n-ary edge that links together all the old cameras and landmarks, acting as a prior on their values thanks to the covariance estimated by the EKF algorithm.

8.2.7.1 Submap Prior Edge

Let consider the set $\mathbf{c}_{old} = \{\mathbf{c}_i\}$, with $i < 0$ of the old cameras, i.e., the cameras poses corresponding to anchor frames initialized in a previous map and shared with the current one. Similarly, lets call $\mathbf{y}_{old} = \{\mathbf{y}_j\}$, $j < 0$ the set of old landmarks. All these elements, i.e., nodes in the graph, will be involved in a unique edge $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}}$ which size is $6n + 3m$, being n the cardinality of the old cameras set and m the cardinality of the old landmarks set. The edge is formulated by considering the error between the mean value estimated in the EKF and the current solution of the optimization. The information matrix associated with this edge comes directly from the covariance estimated by the EKF. This implies that, intuitively, the elements that are properly estimated, i.e., with a low covariance, are more constrained by the edge, while the elements that are poorly observed (i.e., a landmark that was observed a few number of times in the previous map) are more free to be "moved".

Lets split the error vector $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}}$ in $n + m$ elements, where the firsts n elements $(\mathbf{e}_{\mathbf{c}_{-n}}, \mathbf{e}_{\mathbf{c}_{-n+1}}, \ldots, \mathbf{e}_{\mathbf{c}_{-1}})$ represent priors on the camera nodes and the last m elements $(\mathbf{e}_{\mathbf{y}_{-m}}, \mathbf{e}_{\mathbf{y}_{-m+1}}, \ldots, \mathbf{e}_{\mathbf{y}_{-1}})$ represent priors on the landmarks. The prior on a camera node can be decomposed in the translation and orientation error

$$\mathbf{e}_{\mathbf{c}_i} = \begin{bmatrix} \mathbf{e}_{\mathbf{t}_i} \\ \mathbf{e}_{\boldsymbol{\theta}_i} \end{bmatrix}. \tag{8.36}$$

The camera node is composed as $\mathbf{c}_i = \begin{bmatrix} \mathbf{t}_i \\ \mathbf{q}_i \end{bmatrix}$ where

$$\mathbf{t}_i = \mathbf{t}_i^N + \mathbf{t}_i^\delta + \Delta\mathbf{t}_i, \tag{8.37}$$

$$\mathbf{q}_i = \mathbf{q}_i^N \otimes \begin{bmatrix} 1 \\ \frac{\boldsymbol{\theta}_i^\delta}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\boldsymbol{\theta}_i}{2} \end{bmatrix}, \tag{8.38}$$

and the mean values from the EKF estimation are composed by the nominal part and the delta part as

$$\bar{\mathbf{t}}_i = \mathbf{t}_i^N + \overline{\delta\mathbf{t}_i}, \tag{8.39}$$

$$\bar{\mathbf{q}}_i = \mathbf{q}_i^N \otimes \begin{bmatrix} 1 \\ \frac{\overline{\delta\boldsymbol{\theta}_i}}{2} \end{bmatrix}. \tag{8.40}$$

It has to be noticed that the nominal part are exactly the same, since both in the EKF and in the graph optimization they are not modified. It is convenient to express the error between a camera node in the graph and its constraint values obtained by the EKF estimation by transformation composition using homogeneous matrices:

$$\begin{bmatrix} \mathbf{R}\left(\overline{\mathbf{q}}_j\right)^T & -\mathbf{R}\left(\overline{\mathbf{q}}_j\right)^T \overline{\mathbf{t}}_j \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}\left(\mathbf{q}_j\right) & \mathbf{t}_j \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}\left(\overline{\mathbf{q}}_j\right)^T \mathbf{R}\left(\mathbf{q}_j\right) & \mathbf{R}\left(\overline{\mathbf{q}}_j\right)^T \left(\mathbf{t}_j - \overline{\mathbf{t}}_j\right) \\ 0 & 1 \end{bmatrix}. \tag{8.41}$$

Consequently , the rotation part can be expressed as

$$\begin{aligned} \mathbf{q}_e &= \eta\, \overline{\mathbf{q}}_j^* \otimes \mathbf{q}_j & (8.42)\\ &= \eta \begin{bmatrix} 1 \\ -\frac{\overline{\delta \mathbf{t}_j}}{2} \end{bmatrix} \otimes \mathbf{q}_j^{N^*} \otimes \mathbf{q}_j^N \otimes \begin{bmatrix} 1 \\ \boldsymbol{\theta}_j^\delta \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\boldsymbol{\theta}_j}{2} \end{bmatrix} & (8.43)\\ &= \eta \begin{bmatrix} 1 \\ -\frac{\overline{\delta \mathbf{t}_j}}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \boldsymbol{\theta}_j^\delta \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\boldsymbol{\theta}_j}{2} \end{bmatrix}, & (8.44) \end{aligned}$$

and considering the small angles approximation, the orientation error results

$$\begin{aligned} \mathbf{e}_{\boldsymbol{\theta}_j} &= 2\eta\, \mathbf{q}_{e_{\boldsymbol{\theta}}}, & (8.45)\\ \eta &= \frac{1}{\mathbf{q}_{e_w}}, & (8.46) \end{aligned}$$

where $\mathbf{q}_e = [\mathbf{q}_{e_w} \mathbf{q}_{e_{\boldsymbol{\theta}}}^T]^T$.

The translation part of the error results

$$\begin{aligned} \mathbf{e}_{\mathbf{t}_j} &= \mathbf{R}\left(\overline{\mathbf{q}_j}\right)^T \left(\mathbf{t}_j^N + \overline{\mathbf{t}_j} - \mathbf{t}_j^N - \mathbf{t}_j^{\boldsymbol{\delta}} - \Delta\mathbf{t}_j\right) & (8.47)\\ &= \mathbf{R}\left(\overline{\mathbf{q}_j}\right)^T \left(\overline{\mathbf{t}_j} - \mathbf{t}_j^{\boldsymbol{\delta}} - \Delta\mathbf{t}_j\right), & (8.48) \end{aligned}$$

The prior on the landmark is simply given by the difference between the landmark node value and the mean of the landmark value in the EKF:

$$\begin{aligned} \mathbf{e}_{\mathbf{y}_i} &= \mathbf{y}_i^N + \mathbf{y}_i^\delta + \Delta\mathbf{y}_i - \left(\mathbf{y}_i^N + \overline{\delta\mathbf{y}}_i^{FHP}\right) & (8.49)\\ &= \mathbf{y}_i^\delta + \Delta\mathbf{y}_i - \overline{\delta\mathbf{y}}_i^{FHP} & (8.50) \end{aligned}$$

where $\overline{\delta\mathbf{y}}_i^{FHP}$ is the mean value of the landmark estimation, represented by its delta values in the indirect formulation.

The complete edge formulation is given by collecting all the cameras error and landmarks errors in a unique vector

$$\mathbf{e}^P_{\mathbf{c}_{old}, \mathbf{y}_{old}} = \begin{bmatrix} \mathbf{e}_{\mathbf{c}_{-n}} \\ \cdots \\ \mathbf{e}_{\mathbf{c}_{-1}} \\ \mathbf{e}_{\mathbf{y}_{-m}} \\ \cdots \\ \mathbf{e}_{\mathbf{y}_{-1}} \end{bmatrix}. \tag{8.51}$$

This edge has to be accompanied with a $(n+m)\times(n+m)$ information matrix that can be computed through Jacobians propagation as

$$\mathbf{\Omega}_{\mathbf{c}_{old},\mathbf{y}_{old}} = \left(\frac{\partial \mathbf{e}^{P}_{\mathbf{c}_{old},\mathbf{y}_{old}}}{\partial \delta \mathbf{x}} \mathbf{\Sigma}^{P}_{\mathbf{c}_{old},\mathbf{y}_{old}} \frac{\partial \mathbf{e}^{P}_{\mathbf{c}_{old},\mathbf{y}_{old}}}{\partial \delta \mathbf{x}}^{T} \right)^{-1}, \tag{8.52}$$

where the Jacobians of the edge is computed w.r.t. the delta elements of the EKF state vector, i.e., the elements that compose the covariance matrix $\mathbf{\Sigma}_{\mathbf{c}_{old},\mathbf{y}_{old}}$, obtained by selecting the corresponding block of the EKF covariance matrix. It has to be noticed that the Jacobian matrix is sparse, in particular it is block diagonal:

$$\begin{bmatrix} \frac{\partial \mathbf{e}_{\mathbf{c}_{-n}}}{\partial \delta \mathbf{y}^{AF}_{-n}} & & & & & \\ & \ddots & & & & \\ & & \frac{\partial \mathbf{e}_{\mathbf{c}_{-1}}}{\partial \delta \mathbf{y}^{AF}_{-n}} & & & \\ & & & \frac{\partial \mathbf{e}_{\mathbf{y}_{-m}}}{\partial \delta \mathbf{y}^{FHP}_{-m}} & & \\ & & & & \ddots & \\ & & & & & \frac{\partial \mathbf{e}_{\mathbf{y}_{-1}}}{\partial \delta \mathbf{y}^{FHP}_{-1}} \end{bmatrix}. \tag{8.53}$$

The computation of the information matrix associated with the edge take advantage of this sparsity. Moreover, the Jacobians associated with the landmarks $\frac{\partial \mathbf{e}_{\mathbf{y}_{-i}}}{\partial \delta \mathbf{y}^{FHP}_{-i}}$ are the negative identity matrix $(-\mathbf{I})$. It has to be noticed that the computation of the information matrix has to be performed at each iteration of the minimization algorithm, since changes in the nodes values reflects on changes in the covariance matrix.

8.2.7.2 The Resulting Graph

The creation of the prior edge for the elements that comes from a previous map has to be performed in the Algorithm 9 by adding the Algorithm 10 in the "Edges addition" section (after step 15). Moreover, if we are in a pure monocular setup and the map to optimize is not the first, we do not need to fix a landmark to remove the 7th degree of freedom, since the prior edge cover this issue. On the other hand, when odometric information are available an edge between the last old camera pose and the first camera pose of the current map is added.

Lets take a look to the resulting graph of a small SLAM problem in a submap different from the first, i.e., that involves a prior edge, reported in Figure 8.6. Here we suppose that the landmarks $\mathbf{y}_{-2}$ and $\mathbf{y}_{-1}$ have been shared with the previous map. They are respectively referred to the anchor frame represented by $\mathbf{c}_{-2}$ and $\mathbf{c}_{-1}$, that are shared too. The prior edge is created on all these nodes to summarize the lost measurements. Observation of old landmarks in the new maps are treated as usual with ternary measurement edges.

8.3 From Graph Optimization To EKF

In the previous sections it was described how to start an optimization on a graph correspondent to the EKF state vector structure. The missing step, that will be explained here,

Algorithm 10 PriorEdgeCreation

```
1: $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}} \leftarrow \emptyset$
2: **for all** $\mathbf{c}_i$, $i < 0$ **do**
3:     $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}} \leftarrow \begin{bmatrix} \mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}} \\ \mathbf{e}_{\mathbf{c}_i} \end{bmatrix}$
4: **end for**
5: **for all** $\mathbf{y}_i$, $i < 0$ **do**
6:     $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}} \leftarrow \begin{bmatrix} \mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}} \\ \mathbf{e}_{\mathbf{y}_i} \end{bmatrix}$
7: **end for**
8: $\mathcal{E} \leftarrow \mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}}$
```

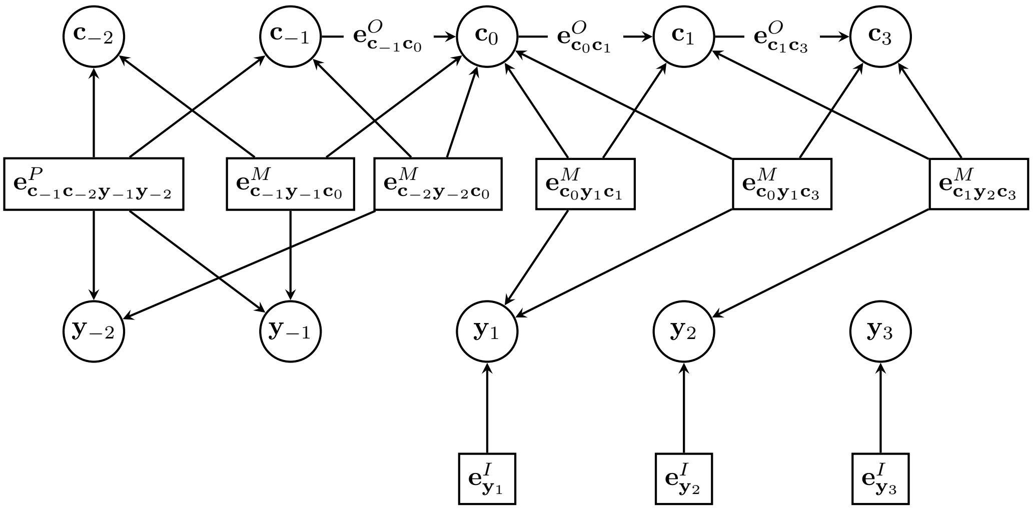

$\mathbf{c}_{-2}$
$\mathbf{c}_{-1}$
$\mathbf{e}^O_{\mathbf{c}_{-1}\mathbf{c}_0}$
$\mathbf{c}_0$
$\mathbf{e}^O_{\mathbf{c}_0\mathbf{c}_1}$
$\mathbf{c}_1$
$\mathbf{e}^O_{\mathbf{c}_1\mathbf{c}_3}$
$\mathbf{c}_3$
$\mathbf{e}^P_{\mathbf{c}_{-1}\mathbf{c}_{-2}\mathbf{y}_{-1}\mathbf{y}_{-2}}$
$\mathbf{e}^M_{\mathbf{c}_{-1}\mathbf{y}_{-1}\mathbf{c}_0}$
$\mathbf{e}^M_{\mathbf{c}_{-2}\mathbf{y}_{-2}\mathbf{c}_0}$
$\mathbf{e}^M_{\mathbf{c}_0\mathbf{y}_1\mathbf{c}_1}$
$\mathbf{e}^M_{\mathbf{c}_0\mathbf{y}_1\mathbf{c}_3}$
$\mathbf{e}^M_{\mathbf{c}_1\mathbf{y}_2\mathbf{c}_3}$
$\mathbf{y}_{-2}$
$\mathbf{y}_{-1}$
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_3$
$\mathbf{e}^I_{\mathbf{y}_1}$
$\mathbf{e}^I_{\mathbf{y}_2}$
$\mathbf{e}^I_{\mathbf{y}_3}$

Figure 8.6: *The graph of a map successive to the first that involves the prior edge* $\mathbf{e}^P_{\mathbf{c}_{old},\mathbf{y}_{old}}$ *on cameras and landmarks initialized in a previous map.*

regards the propagation of the result of the optimization performed on the graph to the Gaussian map memorized in the EKF. The goal of the propagation is to maintain active the CI-SLAM framework mechanism and supply it the new estimations.

8.3.1 Reconstruction Of The Gaussian Map

To reconstruct the Gaussian representation of the map we have to create the mean vector and the covariance matrix of the solution of the optimization. The mean vector can be easily created by appending in a vector $\boldsymbol{\mu}_O$, in the correct order, all the node values at the end of the optimization. We have to take only the proper values of the node (e.g., $\mathbf{c}_i^{\delta}$), while the nominal values (e.g., $\mathbf{c}_i^{N}$) are not changed by the optimization. The covariance matrix $\boldsymbol{\Sigma}_O$ is obtained by the inversion of the Hessian matrix $\mathbf{H}$ of the last iteration, that is the information matrix associated with the solution.

There are some variables that are in the EKF state vector that are not represented by nodes in the graph. For instance, if no odometry is provided, the motion parameters variable $\boldsymbol{\Lambda}$ is in the filter, but not in the graph. Moreover, the last robot pose is in the state filter, but not in the graph and some landmarks can be excluded from the optimization if they have been not measured a sufficient number of times. To propagate new values to the Gaussian map in the EKF we use the same approach of the backpropagation step of the CI-SLAM framework. Calling $\boldsymbol{\mu}_A$ and $\boldsymbol{\Sigma}_{AA}$ respectively the mean and the covariance of the elements that are in the EKF state vector, but not in the optimized graph, $\boldsymbol{\mu}_C$ and $\boldsymbol{\Sigma}_{CC}$ the original mean and covariance of the element in the graph, i.e., their estimations performed with the EKF SLAM algorithm, and $\boldsymbol{\Sigma}_{AC} = \boldsymbol{\Sigma}_{AC}^T$ the covariance block, we can apply the following equations

$$\begin{aligned}
\mathbf{K} &= \boldsymbol{\Sigma}_{AC}\boldsymbol{\Sigma}_{CC}^{-1}, && (8.54)\\
\boldsymbol{\mu}_A^* &= \boldsymbol{\mu}_A + \mathbf{K}\left(\boldsymbol{\mu}_O - \boldsymbol{\mu}_C\right), && (8.55)\\
\boldsymbol{\Sigma}_{AC}^* &= \mathbf{K}\boldsymbol{\Sigma}_O, && (8.56)\\
\boldsymbol{\Sigma}_{AA}^* &= \boldsymbol{\Sigma}_{AA} + \mathbf{K}\left(\boldsymbol{\Sigma}_{AC}^* - \boldsymbol{\Sigma}_{AC}\right)^T, && (8.57)
\end{aligned}$$

to find the new values of the current map elements, that is composed by

$$\begin{aligned}
\boldsymbol{\mu} &= \begin{bmatrix} \boldsymbol{\mu}_A^* \\ \boldsymbol{\mu}_O \end{bmatrix}, && (8.58)\\
\boldsymbol{\Sigma} &= \begin{bmatrix} \boldsymbol{\Sigma}_{AA}^* & \boldsymbol{\Sigma}_{AC}^* \\ {\boldsymbol{\Sigma}_{AC}^*}^T & \boldsymbol{\Sigma}_O \end{bmatrix}. && (8.59)
\end{aligned}$$

8.3.1.1 FIS parameterization

If we are using the FIS parameterization we have to adapt the mean vector $\boldsymbol{\mu}_O$ and the covariance matrix $\boldsymbol{\Sigma}_O$ by removing the estimation about the viewing ray. In particular we have to extract from the mean vector the viewing ray information and to copy them in the scalar values maintained in the FIS parameterization. This allows to correct the initial estimate of the viewing ray direction, that is not continuously estimated in the FIS parameterization. The rows and columns of the covariance matrix corresponding to the first two elements of each landmark have to be deleted, to discard the information on the direction vector that is not represented in the FIS parameterization. After this, Equations 8.54-Equation 8.57 can be applied.

CHAPTER 9

Experimental Results

In this chapter we propose an experimental evaluation of algorithms and techniques presented in previous three chapters. In particular, the parameterizations presented for the standard formulation (presented in Chapter 6) and for the Hybrid Indirect formulation (presented in Chapter 7) of the multi camera EKF-SLAM system are evaluated both in a simulated setup and on real datasets. Framed parameterizations (FHP and FIS), when used in the standard formulation of the EKF SLAM, are not able to perform the loop closure procedure of the CI-SLAM framework due to the use of quaternions. This is verified with a simulated example that demonstrates how the Hybrid Indirect formulation solve this issue. The CIBA SLAM approach (presented in Chapter 8) is then evaluated on simulated and real datasets. All the experiments have been conducted using MoonSLAM, a C++ framework for EKF-SLAM developed during this thesis work. Finally, the time performance of the implemented system are evaluated to validate the capability of the real time execution of the system.

9.1 Parameterization Evaluation

In this section, we present some numerical results showing the behavior of the previously presented parametrizations for Visual EKF-SLAM both in the standard formulation and in the Hybrid Indirect formulation. The simulated data have been obtained in a setup similar to the one proposed in [90] and [93], while real data are taken from the publicly available datasets of the RAWSEEDS[1] project [9].

9.1.1 Evaluation On Simulated Data

The evaluation performed on simulated data, in addition of being an important step during the development process for checking implementation correctness, allows the evaluation of the *consistency* of the state estimation and the analysis on how it is affected by the use of different parameterizations. Formally, an estimate is said to be consistent if it identifies the underlying truth, i.e., if the real value is contained in the estimated probability distribution with a sufficient probability. Since SLAM, in general, is a nonlinear estimation process, no provably consistent estimator can be constructed for it and the consistency of every estimator has to be evaluated experimentally. In particular for the standard EKF-SLAM algorithm, there exists significant empirical evidence showing that the computed state estimates become easily inconsistent [90] [93] [2].

We perform the evaluation of the consistency in a setup similar to the one proposed in [90] [93]: a Montecarlo simulation of a SLAM experiment; the repetition of the same experiment, driven by simulated data with the addition of random noise, is conducted moving the robot on a nominal trajectory and generating image projections from a known map. The motion model of the EKF SLAM takes odometric input corrupted by noise sampled from a Gaussian distribution, thus the camera does not follow the nominal trajectory exactly. The coordinate of point projections on the images are corrupted by Gaussian noise too, thus measurements differ from the nominal values. Alike what done in [90] [93] [10], we are interested in the evaluation of the average Normalized Estimation Error Squared (NEES) [2] during the entire robot trajectory. NEES is evaluated for each time step $k = 0 : T$ as the Mahalanobis distance between the robot true pose $\mathbf{\Gamma}^*_k$ and the estimation of it. The pose estimation is expressed with $\mathbf{\Gamma}^n_k$, $\mathbf{\Sigma}^n_k$, which are respectively the mean values and the covariance matrix at time k during each trial $n = 1 : N$ of the Montecarlo simulation. The rest of the variables in the filter, i.e., the landmark estimations, are not considered, since we assume that if the trajectory is properly estimated, the map is consequently correct, being them estimated simultaneously in SLAM.

The average NEES value at time k is defined by:

$$\bar{\epsilon}_k = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{\Gamma}^n_k - \mathbf{\Gamma}^*_k)^T(\mathbf{\Sigma}^n_k)^{-1}(\mathbf{\Gamma}^n_k - \mathbf{\Gamma}^*_k). \tag{9.1}$$

Consistency is evaluated by checking the 95% confidence interval of $\bar{\epsilon}_t$, which is in turn distributed as a $\chi^2_{N\times 6}$ random variable. In the computation of NEES, quaternions are converted in Euler angles and uncertainty is propagated through this transformation to compute $\mathbf{\Sigma}^n_t$ in Equation 9.1. The conversion of the rotation representation is needed since quaternions covariances matrix are singular, i.e., not invertible as required by Equation 9.1. The filter is considered too optimistic (i.e., errors in camera pose are underestimated) if the NEES value is above the upper bound (H) of the confidence interval , conservative when it is below the lower bound (L) of the confidence interval and consistent if it operates between the two bounds.

We evaluate the average NEES both in the monocular and the stereo setup, running a set of experiments that differ in the followed trajectory, in the noise that corrupts the

[1]http://www.rawseeds.org

motion and in the initialization of the inverse depth of the points (both in the mean value and standard deviation). Points are initialized at equal distance from the camera center. We have verified that when multiple spherical initialization functions for the same parameterization are possible, the results in terms of consistency are very similar.

We propose to evaluate the average NEES by comparing the percentage of time in which the filter operates in the consistent range and in the optimistic range (the remaining percentage represents the conservative behaviour). Moreover, for the time steps in which the filter operates in the optimistic range we evaluate the average inconsistency (AI), i.e., a value computed as

$$AI = \frac{\sum_{t:\bar{\epsilon}_t > H} \bar{\epsilon}_t - H}{T_{\bar{\epsilon}_t > H}}, \tag{9.2}$$

where $T_{\bar{\epsilon}_t > H}$ is the number of frames in which the filter operates in optimistic range and the sum is performed only on steps performed in the optimistic range. The higher the value of the average inconsistency, the more average NEES is far from consistency.

We have to point out that, differently from the setup proposed in [90] and related works, we use parameterizations with shared anchor points/frames, as proposed in Chapter 6. This does not change the estimation results, since the replicated information (i.e., the non shared anchor point/frame) are linearly correlated and the changes on one element reflects on the others. Another difference is in the landmark addition procedure: our system adds all the new landmarks available in an image when their are at least 5, while the original setup add at least one point per image apart in the first image, when 10 landmarks are initialized. Our policy enhances the use of the shared parametrization and we experienced that this has good effects on the consistency of the filter too, if compared with the results obtained with the policy of [90]. The EKF is updated using the iterative procedure explained in Section 5.4.6.1 and in particular using only the first 10 most informative measurements. We have verified that different update schema do not produces significant changes in the NEES evaluation. Inconsistent (i.e., landmarks with negative inverse depth) and unstable landmarks (i.e., landmarks not matched for at least 50% of the time they have been projected on the image plane) are removed from the map. In the stereo case cameras have a distance of 20 cm on the horizontal axis, landmarks are initialized in the left camera and measured in both cameras. For the experiment we simulate a 640×480 camera with a focal length of 320px, corresponding to a $90°$ angle of view. The first two coefficient of the radial distortion function are set to 0.1 , while the third and the tangential distortion parameters are null.

The FIS parameterization represents a special case in the panorama of the inverse depth/scale parameterizations: it does not estimate the direction of the viewing ray and it considers as exact the value provided at initialization, i.e., the output of the salient point detector. This is obviously a weakness in the ability of estimation, since the viewing ray can not be refined with new measurements. To test explicitly this aspect, we repeat the experiments in two different setup: without the random noise addition in the initialization and with a random noise on the first perceived point. Since we expect that the errors introduced by the simplified model proposed for FIS have to decrease growing the camera resolution, we perform experiments also with a high resolution simulated camera (2520×1920).

A last comment on the FIS parameterization is about its measurement model: in Equation 6.66 the noise $\boldsymbol{\delta}_{34}$ is added to the coordinates of the point perceived when the land-

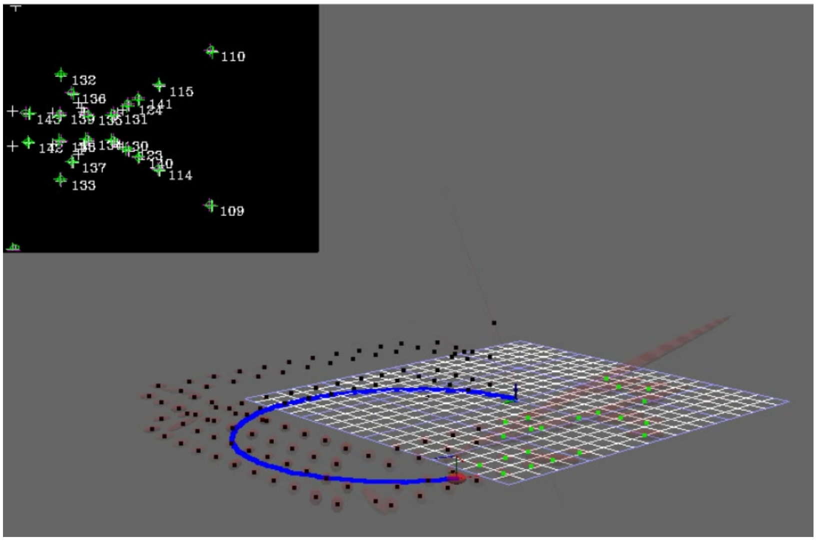


Figure 9.1: *A screenshot of MoonSLAM framework working in the simulated cloister*

mark was initialized. This noise is considered to be independent from the measurement noise (i.e., $\boldsymbol{\delta}_{12}$) and we assume its covariance matrix equal to a 2×2 identity, i.e., with a 1 pixel standard deviation on each coordinate. Differently from other parameterizations, the uncertainty on the initial point can not be reduced, thus the presence of the $\boldsymbol{\delta}_{34}$ element may results in an artificial covariance inflation of the measurement that influences the average NEES analysis. To perform a fair comparison we evaluate the FIS parameterization setting the covariance matrix of $\boldsymbol{\delta}_{34}$ in two different ways: as the identity matrix and as a zero matrix. The latter is equivalent to remove the additional noise from the model and it is referred in the following experiments as FIS0. Moreover, we give en empirical evaluation of the behavior of the covariance estimation for FIS and FIS0 with respect to the others parametrization by plotting the estimate errors and the confidence interval calculated on the estimated covariance in a selected experiment. FHP, FIS and FIS0 are evaluated both in the standard formulation of the EKF SLAM and in the proposed Hybrid Formulation, i.e., with the indirect parameterization named FHP$_i$, FIS$_i$ and FIS0$_i$.

Experiments are performed with a map of 72 points distributed in a cloister-like environment, as the one originally introduced in [90], of 12x12 meters on two different planes, respectively at the height $z = -0.5$ and $z = 0.5$ meters (a screenshot of MoonSLAM framework working in the simulated cloister is shown in Figure 9.1).

Two different nominal motions for the camera are used: the first is a simple planar motion where the robot performs a circular trajectory with a 10 meter diameter on the plane $z = 0$ and the camera points forward in the direction of the motion. The second motion is generated by adding a sinusoidal motion to the radius of the circle. Moreover the robot orientation changes in time: it performs complete rotation around the x axis, i.e., the axis that points in the motion forward direction, and it oscillates around the y and z axis. Figure 9.2 shows the resulting trajectory with some salient robot orientations. The robot performs a complete turn in 1000 steps and 4 complete loops are performed.

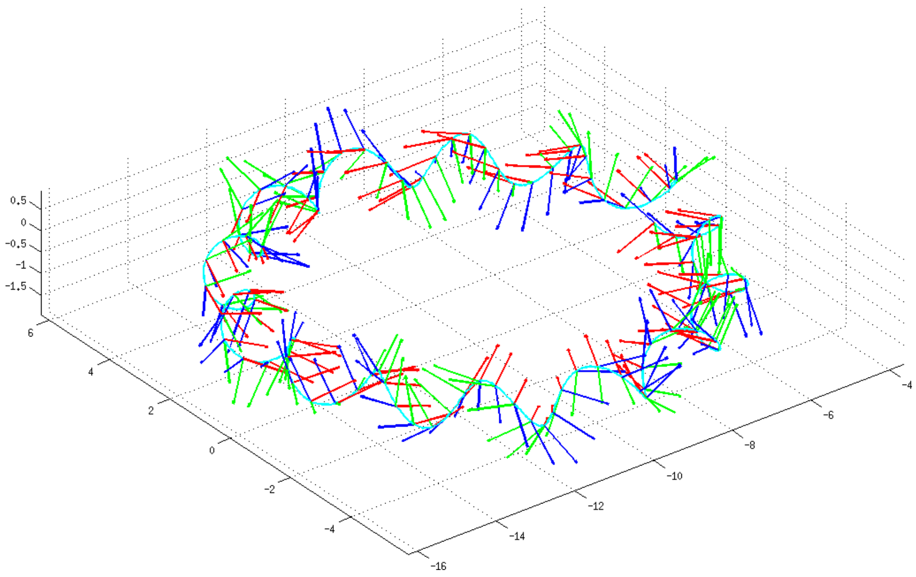


Figure 9.2: *The second trajectory proposed for the simulations. A sinusoidal motion is added to the radius of the circle and the orientation of the robot is varied during the motion around all the axes.*

9.1.1.1 Monocular Simulated Experiments

We performed 7 experiments differing in camera motion and in the noise added it. For each experiment we tested three different initialization depths for the landmarks. Parameters for all these experiments are summarized in Table 9.1, where experiments are identified by progressive numbers followed by a letter to identify different initializations. Letter "*a*" indicates that the initialization is performed with an initial inverse depth of 1m^{-1} with a standard deviation of 1m^{-1}, letter "*b*" indicates initial inverse depth of 0.1m^{-1} (depth 10m) with 0.5m^{-1} standard deviation and "*c*" initializes point inverse depth with 0.01m^{-1} (depth 100m) with 0.5m^{-1} standard deviation. These experiments are repeated in three different setup: in the first one the initial perception of landmark in the image is exact, i.e., the exact viewing ray is used to initialize the features. In the second setup, the initial landmark perception is corrupted by a Gaussian random noise with 1 pixel standard deviation, as done in [90]. In the third repetition we corrupt the initial landmark perception, but a high resolution simulated camera (with 2520×1920 pixel per image) is used.

In Experiments 1 and 2 the robot moves on a plane forward on x axis by 8cm at each time step and rotates of $0.9°$ around the z axis, performing a complete loop in 400 time steps. In Experiment 1 the motion is corrupted with a Gaussian noise with 2.5mm standard deviation on each axis and $0.025°$ around each axis; the covariance matrix of the noise of the motion model is set with the same values used in the random noise generation. In Experiment 2 the noise is halved. Experiment 3 and 4 are performed with halved motion, thus the robot completes a turn of the cloister in 800 steps. In this case, the applied noise is double in Experiment 4 with respect to Experiment 3. The latter has the same noise of Experiment 1. Experiments 5, 6 and 7 use the 3D trajectory of Figure 9.2 and the noise is

Table 9.1: *Experiments setup for the Monocular (between* 1 *and* 7*) and Stereo simulated environment (between* 8 *and* 14*). Experiments differ in motion, noise added on motion, and feature inverse depth/scale initialization*

#	Motion		Noises		Initialization	
	ΔX $[m]$	$\Delta\Psi$ $[°]$	$\Delta\mathbf{t}$ $[mm]$	$\Delta\boldsymbol{\omega}$ $[°]$	ρ $[m^{-1}]$	σ $[m^{-1}]$
1.a, 8.a	0.08	0.9	2.5	0.025	1	1
1.b, 8.b					0.1	0.5
1.c, 8.c					0.01	0.5
2.a, 9.a			1.25	0.0125	1	1
2.b, 9.a					0.1	0.5
2.c, 9.a					0.01	0.5
3.a, 10.a	0.04	0.45	2.5	0.025	1	1
3.b, 10.b					0.1	0.5
3.c, 10.c					0.01	0.5
4.a, 11.a			5.0	0.05	1	1
4.b, 11.b					0.1	0.5
4.c, 11.c					0.01	0.5
5.a, 12.a	3D	3D	1.25	0.0125	1	1
5.b, 12.b					0.1	0.5
5.c, 12.c					0.01	0.5
6.a, 13.a			2.5	0.025	1	1
6.b, 13.b					0.1	0.5
6.c, 13.c					0.01	0.5
7.a, 14.a			5.0	0.05	1	1
7.b, 14.b					0.1	0.5
7.c, 14.c					0.01	0.5

doubled in each experiment, starting from the noise value of Experiment 2.

The results with the exact initial point are reported in Table 9.2 and Table 9.3, respectively for the behavior of the filter and for the mean inconsistency; some average NEES results are plotted in Figures 9.3 to 9.8 allowing visual comparison. For each experiment we highlight in bold the maximum value of the operative time in the consistent range and the minimum value for the optimistic range for IS, UID, AHP, FHP and FHP$_i$ parameterizations. FIS0 and FIS0$_i$ parameterization are treated apart and their values are highlight only when they are better than the others parameterizations, while FIS and FIS$_i$ results are not extensively commented in the next since theirs results are not directly comparable to the others parameterizations due to covariance inflation, as stated in the beginning of this section. In all the experiments IS shows poor performance in NEES, as already shown in [90], [93] and [10][2], i.e., it almost never operates in consistent range, so its results will not be discussed in the following. In all the experiments but 7, we notice that FIS parametrization operates always in consistent or conservative range and it almost never falls in the optimistic case. This result has to be considered as not directly comparable with the other parameterizations, since NEES evaluation on FIS takes advantage from the noise addition in the measurement model, thus it is reported in the last two columns and not directly compared with the others. In the last part of this section we perform a simple analysis of these effects by comparing the covariance estimation on a single run.

In Experiment $1.a$ no one of the parameterization is able to operate in the consistent range for more than the 50% of the time. In particular UID, AHP, FHP and FHP$_i$ show similar performance, approaching the 50% of time in the consistent range. The FIS0 parameterization, and its indirect variant FIS0$_i$ stops at 13% of time in the consistent range, but the analysis of the mean value of inconsistency (Table 9.3) shows that when FIS0 falls in the optimistic range, its mean value of inconsistency is comparable with the other parameterization, i.e., when it is in the optimistic range, it operates very close to the consistency range. IS results the worst parameterization for the consistency: it operates in the consistent range only for the 2% of the time. On the contrary, FIS and FIS$_i$ never fall in the optimistic range and they remains in the conservative range for the most of the time. Since the behavior of IS and FIS parameterizations reflects these results in all the next experiments, we skip hereafter to comment their performances, concluding that IS is the worst parameterization in terms of consistency and FIS consistency values are not directly comparable with the other parameterizations, mainly due to the presence of the additional noise in the measurement model. All these conclusions can be drawn also from the observation of Figure 9.3, which reports the value of the average NEES in the time for each parameterization.

Experiment $1.b$ repeats the Experiment $1.a$ but with a different initialization of the unknown depth of the point, which passes from 1 meter to 10 meters. The change in the initialization greatly improves the consistency: UID, AHP, FHP and FHP$_i$ operates now in the consistent range for more than 90% of time. FIS0 and FIS0$_i$ reach the 80%, showing the greatest improvement from the previous experiment. Moreover, the analysis of the mean inconsistency for this experiment shows that all the parameterization are very close to consistency also when they operates in the optimistic range. The same considerations

[2]The results for FHP here presented differ from the ones presented in [10] because in the article FHP points were initialized on a plane instead of the sphere and this results in a better NEES results that is not directly comparable to the others.

Table 9.2: *Monocular NEES evaluation- Consistency analysis, exact initialization*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	Cons %	2	40	48	48	**49**	13	13	30	29
	Opt %	98	59	52	52	**50**	87	87	0	0
1.b	Cons %	7	**93**	92	**93**	**93**	80	81	32	31
	Opt %	93	**4**	6	5	5	20	19	0	0
1.c	Cons %	3	96	96	**97**	**97**	75	76	23	22
	Opt %	97	3	2	**1**	**1**	25	24	0	0
2.a	Cons %	3	41	47	**49**	48	32	33	60	60
	Opt %	97	59	53	**51**	**51**	68	67	0	0
2.b	Cons %	12	74	81	81	**83**	77	81	46	45
	Opt %	88	26	19	18	**17**	23	19	0	0
2.c	Cons %	4	**80**	76	75	76	73	73	45	44
	Opt %	96	**20**	24	24	24	27	27	0	0
3.a	Cons %	5	**47**	40	45	46	15	14	4	4
	Opt %	95	**53**	60	55	54	85	86	0	0
3.b	Cons %	5	29	**34**	33	**34**	**49**	44	7	7
	Opt %	95	71	66	67	**65**	**51**	56	0	0
3.c	Cons %	5	48	50	52	**54**	49	48	8	9
	Opt %	95	52	50	48	**46**	51	52	0	0
4.a	Cons %	1	3	**4**	**4**	**4**	2	2	78	78
	Opt %	99	97	**96**	**96**	**96**	98	98	5	6
4.b	Cons %	4	**7**	**7**	**7**	**7**	2	2	75	76
	Opt %	96	**93**	**93**	**93**	**93**	98	98	17	16
4.c	Cons %	2	**5**	4	4	**5**	3	3	88	88
	Opt %	98	**95**	96	96	96	97	97	5	5
5.a	Cons %	1	86	77	83	**88**	77	82	18	30
	Opt %	99	14	23	17	**12**	23	18	0	0
5.b	Cons %	2	**98**	97	95	**98**	78	87	13	12
	Opt %	98	**2**	**2**	5	**2**	22	12	0	0
5.c	Cons %	2	**99**	98	98	98	90	88	16	16
	Opt %	98	1	1	**0**	2	10	11	0	0
6.a	Cons %	1	29	30	**31**	27	14	14	37	34
	Opt %	99	71	70	**69**	73	86	86	0	0
6.b	Cons %	2	29	28	31	**44**	19	16	33	43
	Opt %	98	71	72	69	**56**	81	84	0	0
6.c	Cons %	1	46	60	**62**	56	25	17	53	48
	Opt %	99	54	40	**38**	44	75	83	0	0
7.a	Cons %	1	1	**2**	1	1	1	1	25	27
	Opt %	99	99	**98**	99	99	99	99	66	64
7.b	Cons %	0	1	**2**	**2**	**2**	0	0	33	30
	Opt %	100	99	**98**	**98**	**98**	100	100	58	62
7.c	Cons %	0	**1**	**1**	**1**	**1**	0	0	26	32
	Opt %	100	**99**	**99**	**99**	**99**	100	100	71	65

Table 9.3: *Monocular - Mean inconsistency, exact initialization*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	149.3	1.1	**0.8**	**0.8**	**0.8**	1.4	1.4	0.8	0.8
1.b	27.7	**0.3**	0.4	0.4	0.4	0.4	0.4	–	–
1.c	55.2	**0.2**	**0.2**	**0.2**	**0.2**	0.5	0.4	–	–
2.a	110.5	2.0	1.3	1.3	**1.2**	**0.8**	**0.8**	0.0	0.0
2.b	19.1	0.7	**0.6**	**0.6**	**0.6**	**0.5**	0.6	–	–
2.c	39.9	**0.7**	**0.7**	**0.7**	**0.7**	0.6	**0.5**	–	–
3.a	181.6	**0.6**	**0.6**	**0.6**	**0.6**	1.8	1.8	0.7	0.7
3.b	60.1	0.9	**0.7**	**0.7**	**0.7**	0.8	1.1	–	–
3.c	64.3	**0.5**	**0.5**	**0.5**	**0.5**	0.7	0.6	–	–
4.a	908.2	13.1	**12.7**	**12.7**	12.9	13.0	13.3	0.6	0.6
4.b	151.4	11.6	**11.3**	11.4	**11.3**	16.0	15.9	0.4	0.4
4.c	300.7	11.9	11.6	11.6	**11.5**	15.0	15.0	0.4	0.4
5.a	81.8	**0.3**	0.4	0.4	**0.3**	0.4	0.5	0.4	0.4
5.b	12.7	**0.2**	**0.2**	0.3	**0.2**	0.5	0.5	–	–
5.c	25.3	0.3	0.3	**0.2**	**0.2**	0.3	0.4	–	–
6.a	167.0	**1.3**	2.0	1.5	1.7	2.5	2.5	1.8	1.8
6.b	26.5	1.5	2.2	1.6	**1.4**	2.0	2.1	–	0.7
6.c	52.2	1.2	**1.0**	1.3	**1.0**	1.4	2.2	–	–
7.a	858.1	22.7	**22.3**	23.2	22.8	26.2	25.2	4.2	3.7
7.b	230.3	20.0	19.8	**19.3**	22.3	26.3	27.4	3.1	4.4
7.c	527.9	22.8	22.4	22.2	**20.9**	23.9	25.5	5.5	4.4

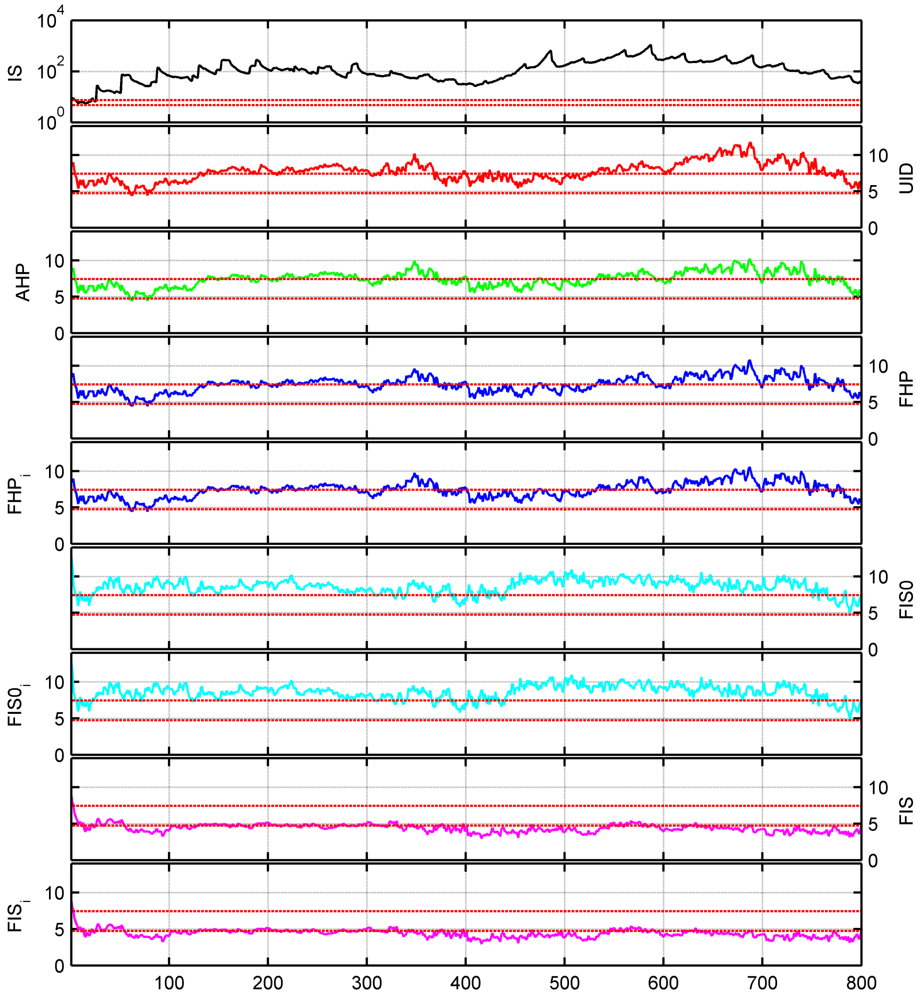


Figure 9.3: *Average NEES values for Experiment 1.a (Monocular) with exact initialization of viewing ray for each parameterization. The confidence interval of consistency is delimited with two horizontal dashed red lines. The IS parameterization performs worst than others and it looses quickly the consistency (notice the ordinate axis is in logarithmic scale). UID, AHP, FHP and FHP*$_i$ *perform very similar, thus differences between theirs curves are not appreciable, apart from a peak on the NEES value of UID parameterization which overtake the value* 10 *just before iteration* 700*; the average NEES is included in the confidence interval for about half of the time and it remains close to the consistency range also when the NEES value is over the highest threshold. FIS*0 *and FIS*0_i*, i.e., the Framed Inverse Scale parameterization without the introduction of Gaussian noise in the measurement equation on the initial viewing ray, show substantially equal performance and they perform slightly worst than UID, AHP and FHP; although they operates for the most of the time in the optimistic range, they are quite close to consistency. The FIS and FIS*$_i$ *curves are substantially equal too, but they evidence that noise in the measurement equation gives better performance in average NEES: the curve is below the lower limit of the confidence interval for the most of the time.*

can be applied to the results of Experiment $1.c$, where the initial depth is increased to 100 meters. In this case, FIS0 and FIS0$_i$ worsen a little bit their performances, but they rest consistent for the most of the time.

Experiments $2.a$, $2.b$ and $2.c$ use the same motion speed of the relative experiments of set 1, but the additional noise standard deviation is halved. The analysis of the consistency shows that the change in the noise does not affect so much the performance of UID, AHP, FHP and FHP$_i$, while it affects much more FIS0 and FIS0$_i$. Considering Experiment $2.a$ we notice that only FIS0 and FIS0$_i$ parameterizations improve their performance from a 13% of time in the consistent range to more than 33%. The performance of UID, AHP, FHP and FHP$_i$ degrade in Experiments $2.b$ and $2.c$ about, respectively, 10 and 15 percent points with respect to Experiments $1.b$ and $1.c$. FIS and FIS0 performance remains more similar to the results of Experiments $1.b$ and $1.c$, approaching the performance of other parameterizations. Moreover, the analysis of the mean inconsistency value shows that FIS0 and FIS0$_i$ are more close to the upper bound of the consistency range than other parameterizations, especially in Experiment $2.a$. This can be observed in Figure 9.4 too, where the average NEES value of each parameterization for Experiment $2.a$ are plotted.

From these two experiments we can draw a first conclusion: all the parameterizations suffers initialization of the unknown depth close to the camera and, in particular, the FIS0 and FIS0$_i$ parameterizations are especially affected by this problem.

In Experiments 3 and 4 the motion speed is halved with respect to the first two experiments, resulting in a 800 step complete loop of the cloister. In Experiment $3.a$ all the parameterizations maintain similar performances to the one obtained in Experiments $1.a$ and $2.a$, approaching the 50% of time in the consistent range for UID, AHP, FHP and FHP$_i$ parameterizations. Differently from what happens in Experiments $1.b$, $1.c$, $2.b$ and $2.c$, in Experiments $3.b$ and $3.c$ the change in the initialization does not reflects in improvement in consistency and this is especially evident for UID, AHP, FHP and FHP$_i$ parameterization, while FIS0 and FIS0$_i$ gain consistency and they result the more consistent parameterizations in Experiment $3.b$, which is shown in Figure 9.5. In Experiment 4 the noise is doubled with respect to Experiment 3 and, in this case, no parameterization is able to operate in the consistent range (see Figure 9.6). The analysis of the mean value of inconsistency shows that in this case FIS0 performs a little bit worst than others.

Experiments 5, 6 and 7 use the complex motion in 3D from Figure 9.2. In Experiment 5, the one with the lower value of noise added to the motion, all the parameterizations show a consistent behavior for the most of the time and FIS0 confirms to suffer from closest initialization, but it has a quite low value of mean inconsistency, i.e., not very far from the consistent behavior. In Experiment 6 the noise is doubled and in this case the FHP and FHP$_i$ parameterizations perform better than others. In Experiment 7 the noise is doubled once more and parameterizations loose their consistency with a high value of average inconsistency. Differences in performances in Experiment $6.c$, where the filter operates for the most in the consistent range, and $7.a$, where it operates in optimistic range, can be appreciated by the comparison of Figure 9.7 and Figure 9.8.

These experiments are run a second time adding the corruption of the initial point. The complete results are reported in Appendix A in Table A.2 and Table A.3; here we give only a general comment on the experiments. As expected, both FIS and FIS0 performs worst than others: they never operate in the consistency range, while other parameterizations perform a little worst than in the first set of experiments due to the inexact initialization of

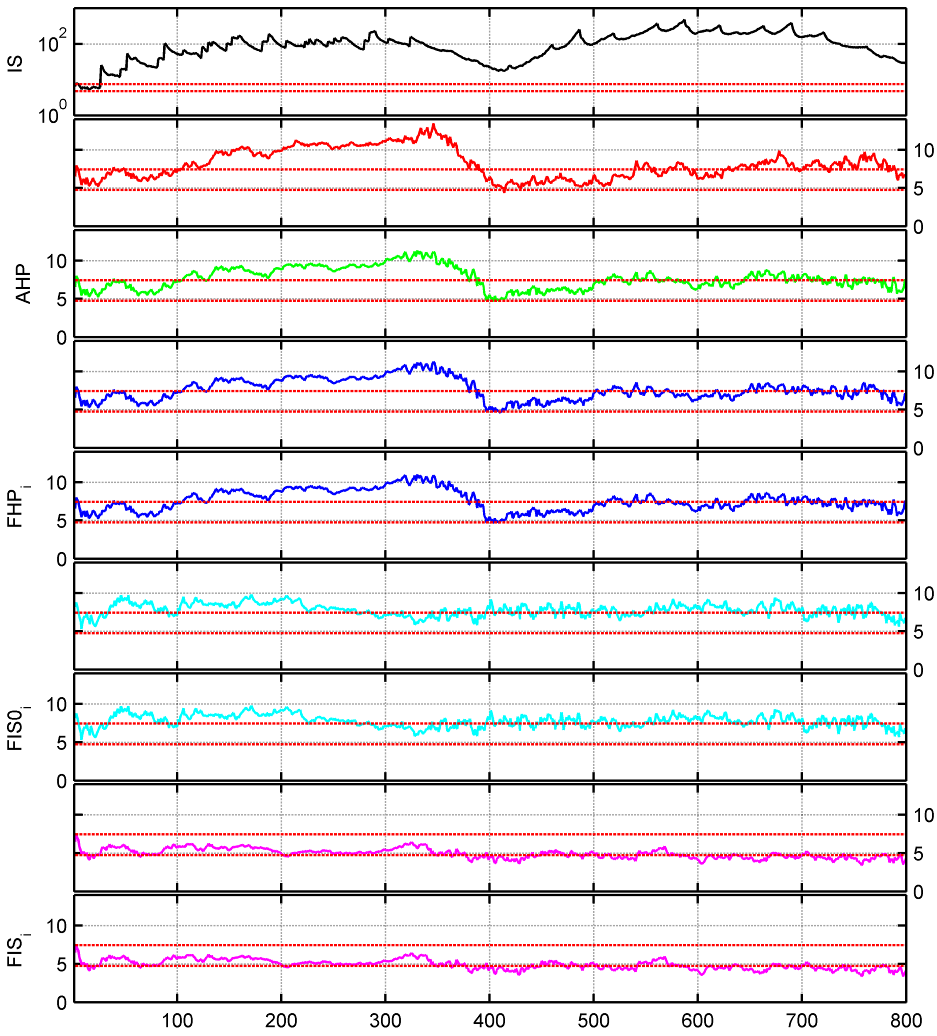


Figure 9.4: *Average NEES values for Experiment 2.a (Monocular) with exact initialization of viewing ray. The results of IS parameterization are the worst: the consistency is not reached for the most of the time and only at frame* 400*, where a complete turn is performed, the average NEES values for IS approaches the upper bound of the confidence interval; notice that ordinate axis is in logarithmic scale for IS. Alike the previous case, all the other parameterizations work in the consistent range for about the half of the time. UID, AHP, FHP and FHP*$_i$ *loose consistency between steps* 100 *and* 400 *and this is more evident for UID, which presents a peak at about step* 350*. They re-enter in the consistent range after step* 400*, where a turn is completed. FIS*0 *and FIS*0_i *results are substantially equal, the same happens for FIS and FIS*$_i$*, meaning that the use of indirect parameterization does not introduces significant differences in average NEES. FIS tends to be close to the lower bound the consistency and to fall below this level, i.e., to perform in the conservative range; FIS*0 *is close to the upper bound and operates in the consistent and optimistic range but it does not suffer of big inconsistency in the first loop of the cloister, i.e. before frame* 400*, as it happens for other parameterizations.*

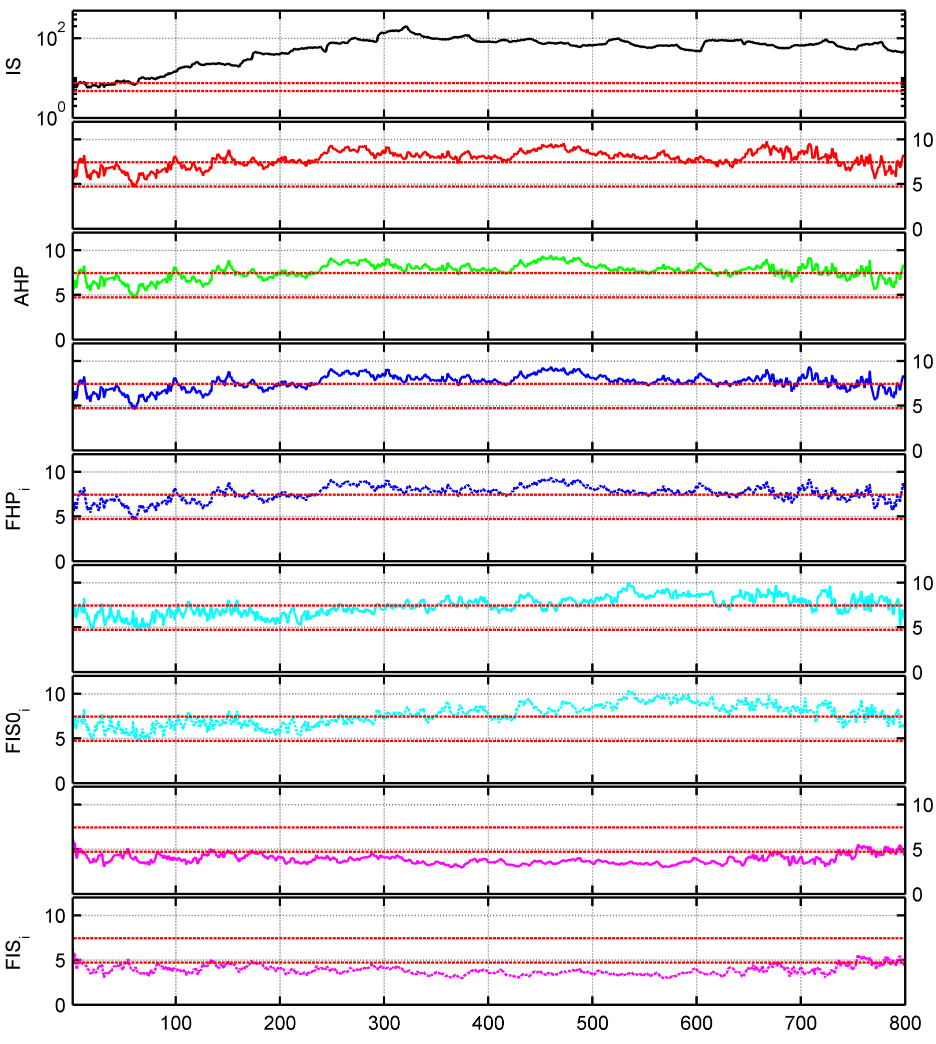


Figure 9.5: *Average NEES values for Experiment 3.b (Monocular) with exact initialization of viewing ray. The results of IS parameterization are the worst: the consistency is loosed before frame* 100*; notice that ordinate axis is in logarithmic scale for IS. UID, AHP, FHP and FHP*$_i$ *show very similar performance; they maintain consistency for about* 30% *of the time. FIS*0 *and FIS*0_i *curves are substantially equal, the same happens for FIS and FIS*$_i$*, meaning that the use of indirect parameterization does not introduces differences in average NEES. FIS performs in the conservative range, i.e., under the lower bound of the confidence interval for the most of the time; FIS*0 *and FIS*0_i *operate in optimistic range for longer time (about* 50%*) than UID, AHP and FHP parameterizations, resulting the parameterizations with the best performance in the consistency analysis.*

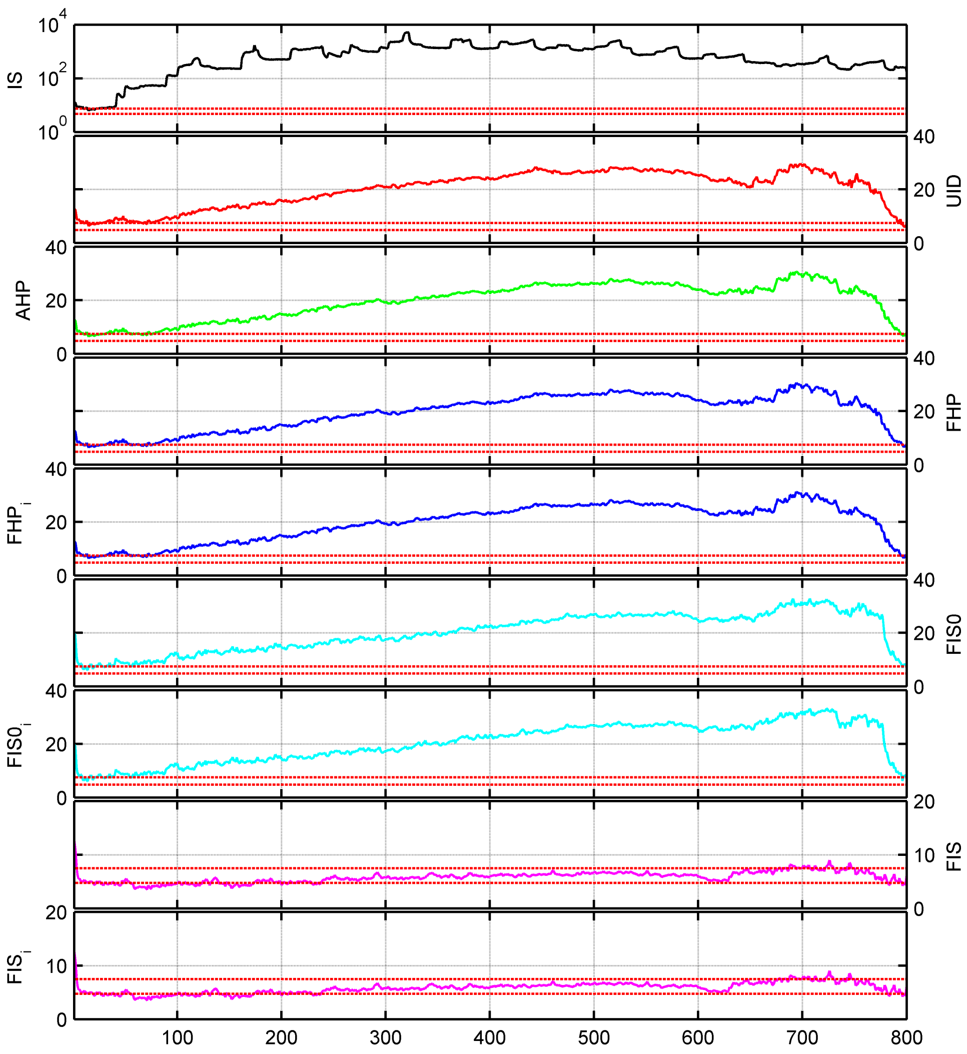


Figure 9.6: *Average NEES values for Experiment 4.a (Monocular) with exact initialization of the viewing ray. The results of IS parameterization are the worst: the consistency is loosed before frame* 100*; notice that ordinate axis is in logarithmic scale for IS. AHP, FHP and FHP$_i$ show very similar performance, while UID is slightly different (e.g., between steps* 600 *and* 700*). FIS*0 *and FIS*0$_i$ *curves are substantially equal, the same happens for FIS and FIS$_i$, meaning that the use of indirect parameterization does not introduces significant differences in average NEES. All the parameterizations but FIS and FIS$_i$ loose consistency quite early. FIS*0 *and FIS*0$_i$ *perform slightly worst, especially around time step* 600*, where their average NEES is the highest. FIS and FIS$_i$ maintain consistency in almost all the experiment, revealing the good effects on the NEES of the noise added in the measurement equation.*

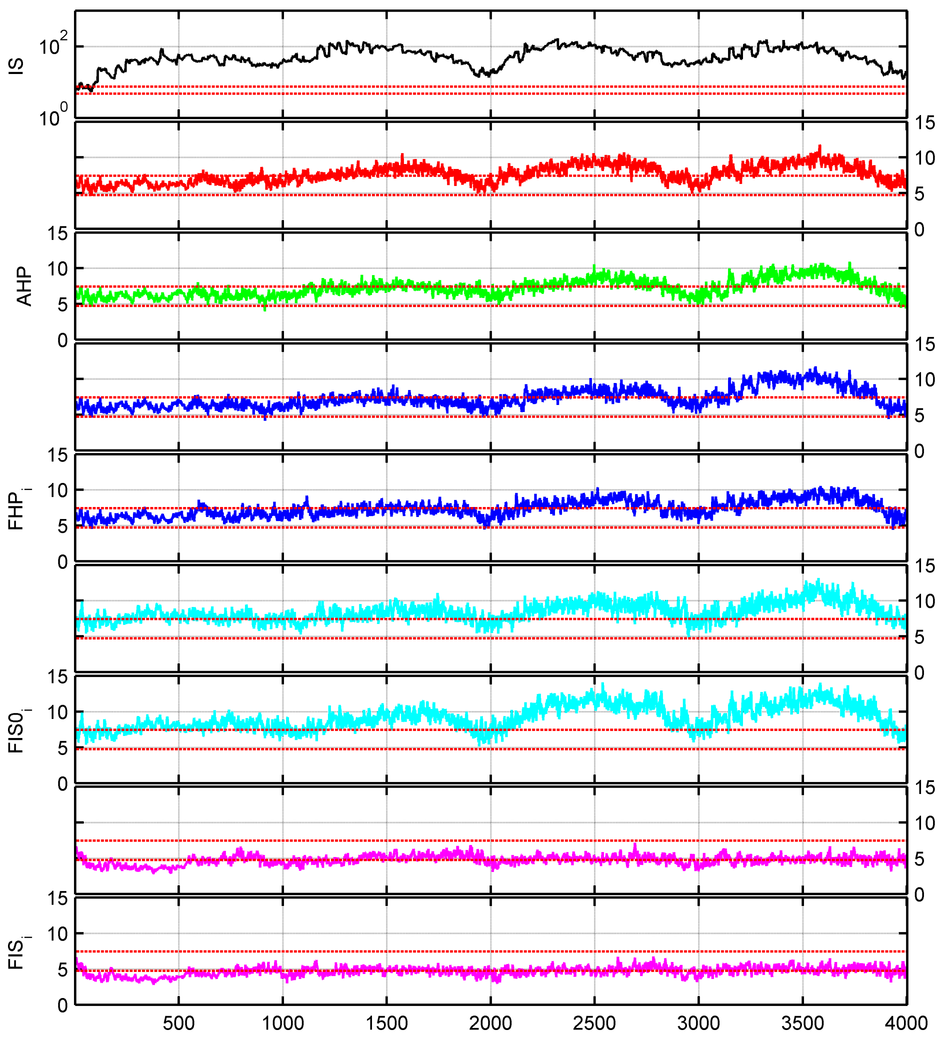


Figure 9.7: *Average NEES values for Experiment 6.c (Monocular) with exact initialization of viewing ray. The motion in this case is not planar, but it follows a 6DoF trajectory. The results using IS parameterization are the worst: the consistency is loosed before frame* 100*; notice that ordinate axis is in logarithmic scale for IS. AHP and FHP show very similar performance, while UID performs slightly worse (e.g., between steps* 2000 *and* 3000*). FHP shows the best performances, although differences with others parameterizations are very small. FIS*0 *and FIS*0_i *differences are very small; these parameterizations operate at the limit of consistency range between frame* 0 *and* 2000*, while they exit from the consistent range after step* 2000*. FIS and FIS*$_i$ *are similar too but the noise addition in the measurement equations guarantee consistency: they operates at the limit of the conservative behavior.*

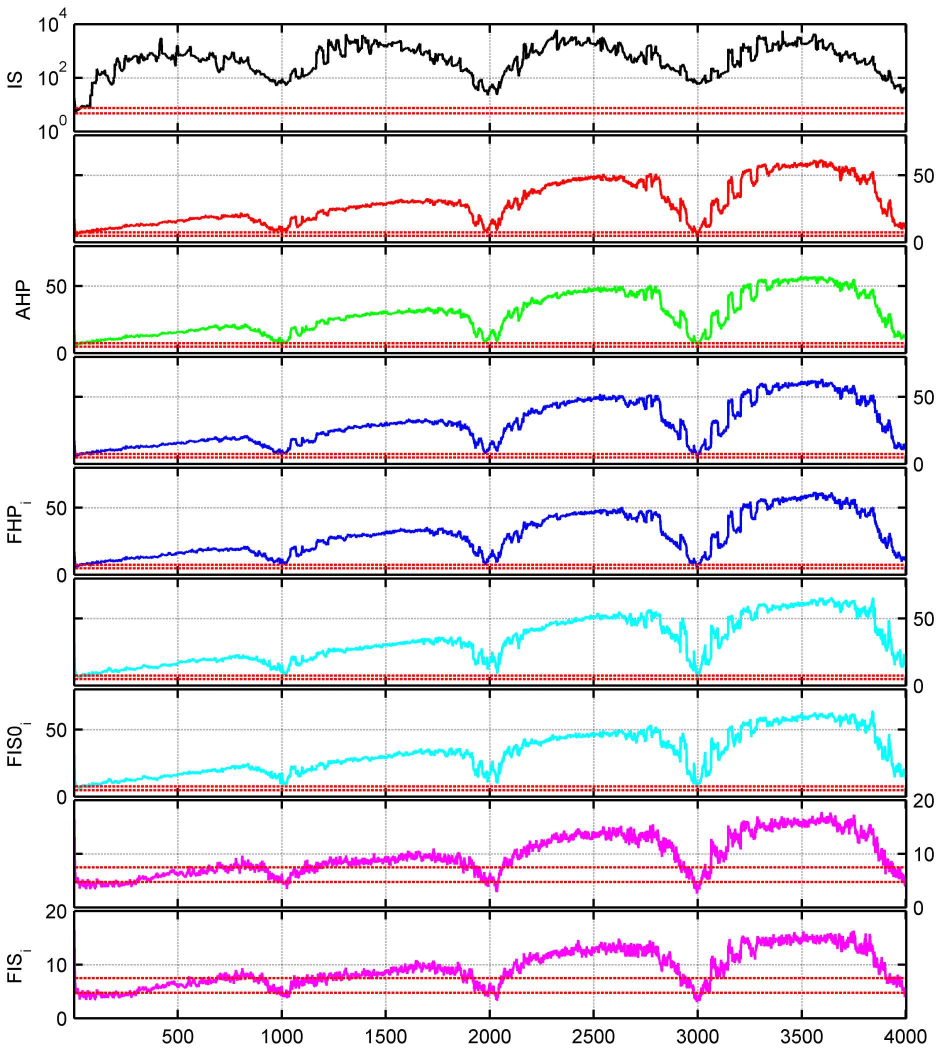


Figure 9.8: *Average NEES values for Experiment 7.a (Monocular) with exact initialization of viewing ray. The motion in this case is not planar, but it follows a 6DoF trajectory. The results of IS parameterization are the worst: the consistency is loosed very early; notice that ordinate axis is in logarithmic scale for IS. UID, AHP, FHP, FHP*$_i$*, FIS*0 *and FIS*0_i *parameterizations show similar performance, but the NEES value is very far from the consistency range. FIS and FIS*$_i$ *have the better performance and they loose consistency only after step* 1000.

the viewing ray.

A third run of the experiment is performed with the high resolution camera. Complete results on the consistency analysis are reported in Appendix A with Table A.4 and Table A.5. The performance in NEES of all the parameterizations but FIS (in all its versions) are worse than the performance obtained with the low resolution camera. This can be explained considering that the high precision in the measurement process has negative effects on the correction of the state performed by the update step of the EKF. This is observed in [53], where it is stated that the larger the covariance matrix of measurement noise, the less the effect on inconsistency. In our case the covariance matrix is always the same, but the change in the camera resolution corresponds to a lower covariance in the measurement, since the higher resolution in the image corresponds to higher angular resolution in the viewing ray. The average NEES values of FIS parameterizations do not degrade with the high camera resolution setup, thus FIS performances results more similar to others parameterizations, although they are far from operate in the consistency range. Figure 9.9 shows the comparison of the average NEES values for the standard camera and the high resolution camera in Experiment $1.c$, while Figure 9.10 shows the same comparison for Experiment $6.a$.

To summarize the analysis of consistency for the monocular experiments we calculate the mean percentage of time in which the filter operates in the optimistic range and the total value of the average inconsistency for all the experiments in the three different proposed setup (i.e., with exact initialization, with noised initialization and with noised initialization using the high resolution camera). To compute the mean time in which the filter operates in the optimistic range we weight the optimistic percentage of each experiment with the experiment length. In Table 9.4 the summarized results are reported within an average value over all the three different setup; these results are plotted in Figure 9.11 too. AHP, UID, FHP and FHP$_i$ perform substantially equal, with a very small advantage for FHP$_i$. IS and FIS (in all its versions) perform bad, being optimistic for the most of the time (obviously this is not true for the FIS with exact initialization). The reason for the poor performances in IS are in absence of the anchor point/frame in the parameterization. FIS poor performance are due to the wrong assumption on the exactness of the initial viewing ray. The total value of the average inconsistency is computed by summing up the values of the average inconsistency for all the experiments in the three different setup. Results are reported in Table 9.5 and plotted in Figure 9.12. They show that UID, AHP, FHP and FHP$_i$ parameterization have similar value of mean inconsistency, i.e., when they are optimistic they reach similar values of NEES. IS performs very bad: its average inconsistency is more than 100 times higher than the better parameterizations. FIS parameterization performs definitely worse than UID, AHP, FHP and FHP$_i$ but has the good property to limits inconsistency when using high resolution cameras: the average value of inconsistency is more than 5 time higher for FHP passing from standard to high resolution camera, while it is only doubled for FIS.

We perform here also a different comparison between parametrizations; we take a single run of the first experiment and we analyze the estimation error for each coordinate and angle around axis with respect to the ground truth considering the $\pm 3\sigma$ confidence interval of the filter standard deviation. The aim of this comparison is to evaluate the impact of the additional noise $\boldsymbol{\delta}_{3,4}$ in FIS parametrization. Results are shown in Figure 9.13. As expected, IS underestimates the covariance and makes large errors. FHP, AHP and UID show

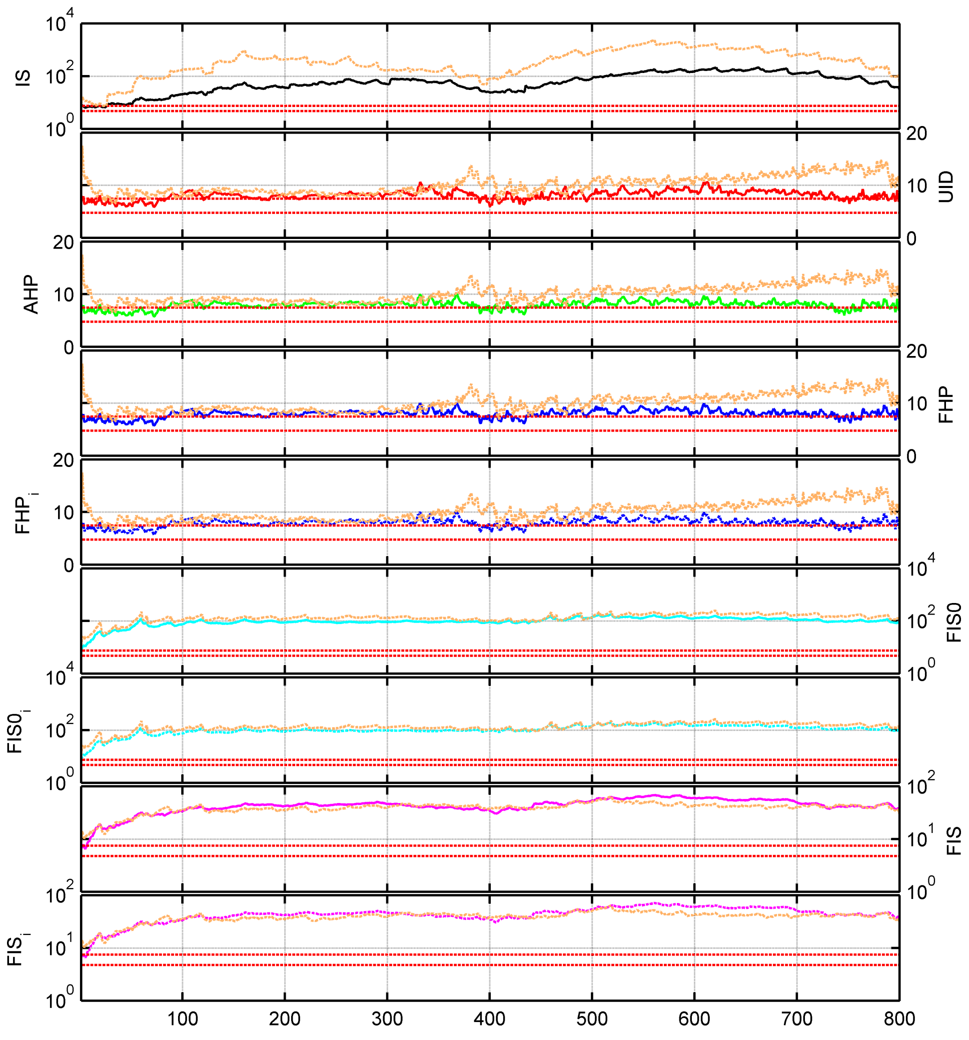


Figure 9.9: *Average NEES values for the parameterizations with noised initialization in Experiment 1.c. The plots compare the standard resolution camera (640 × 480 pixels) with the high resolution camera (2520 × 1920 pixels). Solid lines with same colors of the previous plot are used for the standard camera, while dashed orange lines are used for the high resolution camera. It is noticeable that for all the parameterizations but FIS (in all its versions) the average NEES value increases with in the high resolution camera setup (orange lines). For FIS this is less evident, although FIS is still far from consistency.*

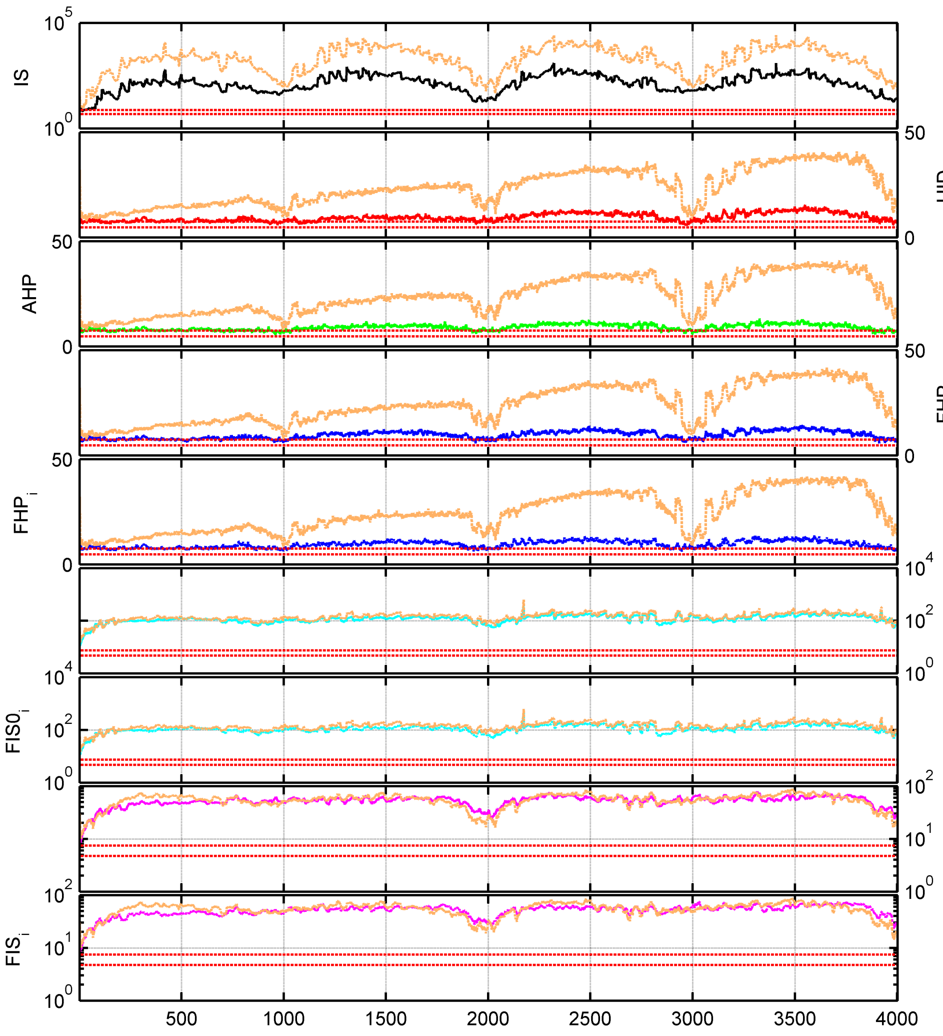


Figure 9.10: *Average NEES values for the parameterizations with noised initialization in Experiment 6.a. The plots compare the standard resolution camera (*640×480 *pixels) with the high resolution camera (*2520×1920 *pixels). Solid lines with same colors of the previous plot are used for the standard camera, while dashed orange lines are used for the high resolution camera. It is noticeable that for all the parameterizations but FIS (in all its versions) the average NEES value increases with in the high resolution camera setup (orange lines), resulting in a behavior that is quite far from the consistency range. FIS performance are slightly affected by the use of the high resolution cameras, thus its performance are more close to the results obtained by others parameterizations.*

Table 9.4: *Summary of results for the mean time the filter operates in the optimistic range. Weighted average is performed on the percentage of operative time in the optimistic range from Tables 9.2, A.2 and A.4 with the experiment length (800 for Experiments 1.a to 4.c, 4000 for Experiments 5.a to 7.c). These results are graphically shown in Figure 9.11. In bold the parameterization that reach the best results for the three different setup and the overall best. Notice that FIS parameterization, in all its versions, is not compared to others parameterization, since it performs very well with the exact initialization and very bad with the noised initialization, due to its particular formulation.*

	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
Exact init.	97.9	55.8	55.0	54.0	**53.0**	65.1	64.8	17.6	17.2
Noised init.	99.6	78.6	78.8	78.5	**74.9**	100	100	99.9	99.9
Noised init. HD	99.9	**93.1**	93.3	**93.1**	**93.1**	100	100	100	100
Mean	99.0	75.8	75.7	75.2	**73.7**	88.4	88.3	72.5	72.4

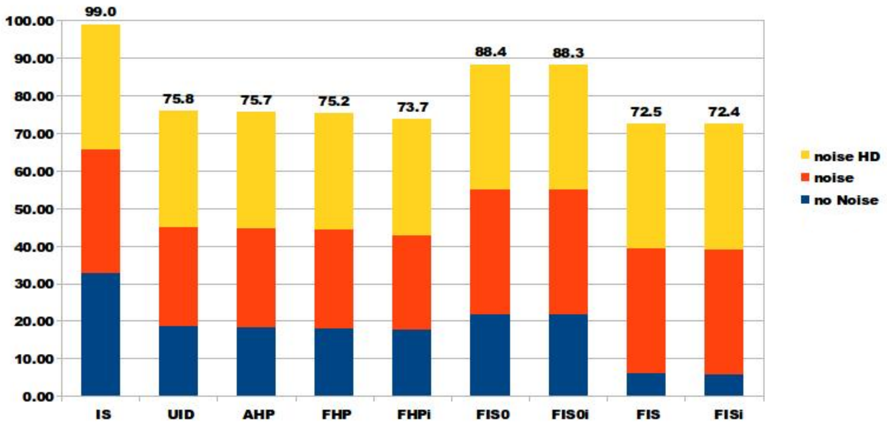


Figure 9.11: *Graphical summary of results for the mean time in which the filter operates in the optimistic range of Table 9.4 in the three different setup (i.e., with exact initialization, with noised initialization and with noised initialization using the high resolution camera). The bar height corresponds to the mean value reported in the table; each bar is partitioned in the value of the three different setups.*

Table 9.5: *Summary of results of the total of average inconsistency (AI) value. The total inconsistency is calculated by summing the values of average inconsistency of Tables 9.3, A.3 and A.5. In bold the parameterizations that reach the best results for the three different setup and the overall best. Notice that FIS parameterization, in all its versions, is not compared to other parameterizations, since it performs very well with the exact initialization and very bad with the noised initialization, due to its particular formulation.*

	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
Exact init.	3182	114	**112**	**112**	**112**	135	138	18	18
Noised init.	4529	125	**121**	123	123	2961	2995	1209	1190
Noised init. HD	92694	630	624	625	**620**	6105	6086	1243	1242
Total	100405	869	857	860	**855**	9201	9220	2470	2450

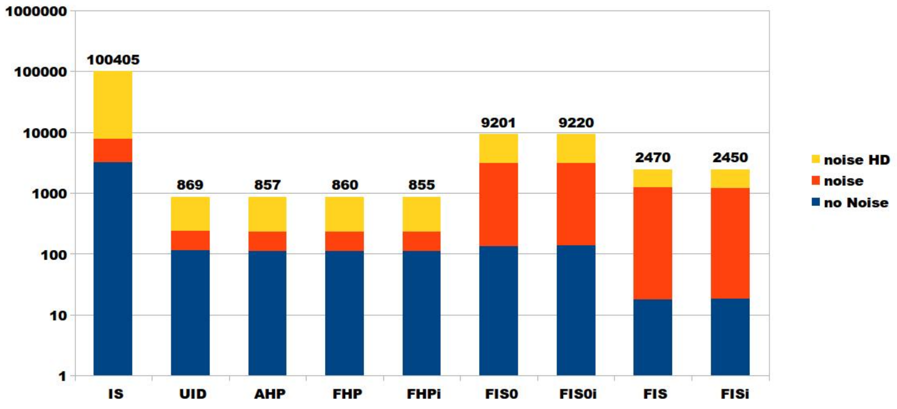


Figure 9.12: *Graphical summary of results of the total of average inconsistency value Table 9.5 in the three different setup (i.e., with exact initialization, with noised initialization and with noised initialization using the high resolution camera). The bar height corresponds to the mean value reported in the Table 9.5; each bar is partitioned in the value of the three different setups, but it has to be noticed that the logarithmic axis on ordinate prevents comparisons between values of parameterizations that performs significantly different.*

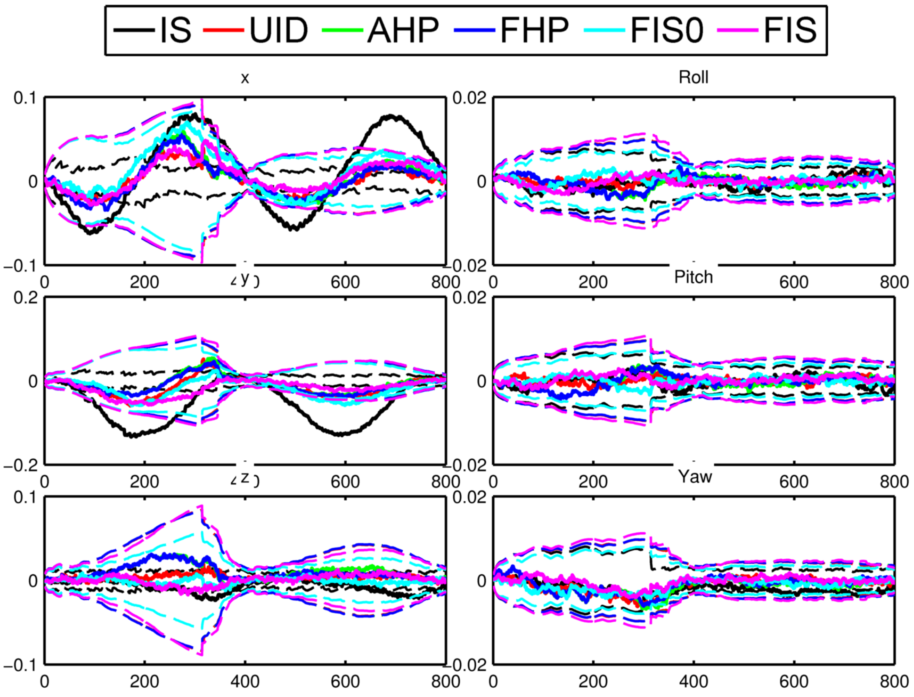


Figure 9.13: *Estimation errors (continuous lines) and estimated standard deviation (dashed lines) of a single run of Experiment 1. UID (red), AHP (green) and FHP (blue) standard deviation estimate are very similar, thus their dashed curves overlaps; IS and FIS*0 *(cyan) underestimate the standard deviations compared to UID, AHP and FHP, while FIS (magenta) overestimate a bit the standard deviations, evidencing the benefits of the addition of the noise in the measurement model. It is noticeable that IS inconsistency is more evident in the translation part, while are comparable to the other parameterization in the rotation part.*

a very similar behaviour while FIS$_0$ underestimates covariance with respect to them. FIS shows very similar covariances with respect to other parameterizations (except IS), confirming that the choice of 1px as standard deviation of additional noise $\boldsymbol{\delta}_{3,4}$ is appropriate and it does not favour FIS too much.

9.1.1.2 Stereo Simulated Experiments

All the experiment performed for the monocular case are repeated in a stereo setup too to evaluate the proposed multi-camera SLAM system. These experiments are referred with numbers from 8 to 14 in Table 9.1. Table 9.6 reports the complete results of the consistency analysis for the case in which the initial viewing ray is exact, while the average inconsistency (AI) values are reported in the Appendix A in Table A.1; results for the case in which the viewing ray is corrupted with noise and for the case in which the viewing ray is corrupted but a high resolution camera is used are reported respectively in Tables A.6, A.7 and Tables A.8, A.9 in the Appendix A.

Table 9.6: *Stereo - Consistency analysis, exact initialization*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	Cons %	0	2	**3**	**3**	**3**	8	**9**	76	74
	Opt %	100	99	**97**	**97**	**97**	**92**	**92**	2	2
8.b	Cons %	16	93	**97**	**97**	96	82	77	4	4
	Opt %	84	7	**3**	**3**	4	19	23	0	0
8.c	Cons %	8	92	**97**	**97**	95	78	71	3	3
	Opt %	92	8	**3**	**3**	5	22	29	0	0
9.a	Cons %	1	**7**	6	6	6	**13**	12	45	43
	Opt %	99	**93**	94	95	94	**87**	88	0	0
9.b	Cons %	17	**97**	92	92	91	51	51	16	19
	Opt %	83	**3**	8	8	9	49	49	0	0
9.c	Cons %	8	**93**	91	**93**	**93**	66	65	15	18
	Opt %	93	8	9	8	**7**	34	35	0	0
10.a	Cons %	0	0	1	1	**3**	**19**	17	28	24
	Opt %	100	100	99	99	**97**	**82**	83	2	2
10.b	Cons %	10	**97**	**97**	**97**	**97**	56	57	13	12
	Opt %	90	**2**	3	**2**	**2**	44	43	0	0
10.c	Cons %	4	**95**	94	**95**	94	78	77	12	12
	Opt %	96	**4**	5	**4**	5	22	23	0	0
11.a	Cons %	0	2	2	2	**3**	**19**	17	61	62
	Opt %	100	**98**	**98**	**98**	**98**	**81**	83	1	1
11.b	Cons %	7	**95**	**95**	**95**	94	80	82	6	6
	Opt %	93	**2**	3	**2**	4	20	18	0	0
11.c	Cons %	2	**94**	90	91	93	82	78	6	6
	Opt %	98	**4**	8	7	5	18	22	0	0
12.a	Cons %	0	0	1	1	**2**	53	**66**	9	10
	Opt %	100	100	99	99	**98**	47	**34**	0	0
12.b	Cons %	36	93	88	86	**96**	80	82	8	8
	Opt %	64	6	11	14	**4**	20	18	0	0
12.c	Cons %	3	87	96	**97**	79	71	73	9	6
	Opt %	97	13	**2**	3	21	29	27	0	0
13.a	Cons %	**0**	**0**	**0**	**0**	**0**	49	**63**	10	11
	Opt %	**100**	**100**	**100**	**100**	**100**	51	**37**	0	0
13.b	Cons %	14	72	**83**	74	73	45	56	9	12
	Opt %	86	28	**17**	25	26	55	44	0	0
13.c	Cons %	2	65	**67**	62	56	52	51	16	13
	Opt %	98	35	**32**	38	43	48	49	0	0
14.a	Cons %	**0**	**0**	**0**	**0**	**0**	**52**	46	15	30
	Opt %	**100**	**100**	**100**	**100**	**100**	**48**	54	1	1
14.b	Cons %	12	80	**83**	80	65	44	48	34	27
	Opt %	88	19	**17**	20	35	56	52	0	0
14.c	Cons %	3	**56**	46	51	39	**60**	40	27	37
	Opt %	97	**44**	54	49	61	**40**	60	0	0

The comparison of Table 9.6 and Table 9.2, i.e., consistency for the stereo setup and the monocular with exact initialization, evidences that the use of the multi camera setup tends to increase inconsistency when close initialization is used and to improve the consistency with far initialization. For instance, Experiment $1.a$ (monocular) shows consistency for about 50% of the time, while its corresponding Experiment $8.a$ (in stereo setup) is clearly inconsistent. The same happens for $2.a$ and $9.a$, evidencing also that a close initialization of the unknown depth of landmarks may be dangerous. For a more detailed analysis, let first consider only the UID, AHP, and FHP (both in the standard and in the indirect formulation) parameterizations. We can affirm that far initialization improves the consistency in the stereo setup: considering the couple of Experiments $3.c$ and $10.c$ we notice that in the stereo setup the filter operates in the consistent range in almost the entire range, being optimistic only in about the 5% of the cases. Experiments $11.b$ and $11.c$, compared with $4.b$ and $4.c$, show significant increment in the consistency for the stereo setup with respect to the monocular case which is optimistic for the most of the time.

The analysis of the Experiments 12 to 14, i.e., the ones with the complex motion in 3D, confirms that the use of the stereo camera increases greatly the consistency with respect to the monocular case if the landmarks are initialized quite far from the camera (with 10 meters and 100 meters initialization). For instance, Experiment $14.b$ shows good consistency behavior with respect to $7.b$ which is almost never consistent. Comparison between Experiments $12.a$ and $5.a$ reveals one more time that close initialization is dangerous: the monocular case shows consistency while the stereo setup loose consistency. The FIS0 parameterization shows a different behavior: in monocular case it tends to loose consistency when points are initialized close to the camera, while in the stereo setup it shows good performance in some of these cases, as in Experiments $12.a$, $13.a$ and $14.a$, where it performs better than other parameterizations. In particular, in these experiments, FIS0 is the unique parameterization that reaches a good level of consistency. On the other side, FIS0 suffers far initializations in the stereo setup in other experiments.

The addition of random noise on the initial viewing ray (Table A.6 and Table A.7 in the Appendix A) does not affect performance in consistency, although we have an obvious degradation of the NEES average value. As expected, FIS parameterization performs worse, since it is not able to correct the initial direction of the viewing ray. The use of the high resolution camera has bad effects on the consistency, as already observed for the monocular case, reaching a complete inconsistent behavior in almost all the experiments. For detailed results in this case see Table A.8 and Table A.9 in the Appendix A.

IS parameterization was never considered up to here since it performs worse than others, but in a stereo setup there are some cases in which it is quite consistent: this is the case of Experiment $12.b$ and $13.b$, in which it performs similarly to the other parameterizations when the initial viewing ray is initialized without noise addition. Comparison between the three runs of Experiment $12.b$, i.e., with the exact initialization, the noised initialization and the noised initialization in the high resolution camera setup, are shown for a visual comparison respectively in Figure 9.14, Figure 9.15 and Figure 9.16.

We summarize the results obtained with the stereo setup in the same way we do for the monocular case, computing the mean time in which the filter operates in optimistic range and the total value of mean inconsistency in the three different setup. Results are reported in Table 9.7 and Table 9.8 and plotted respectively in Figure 9.17 and Figure 9.18. In this case, the AHP parametrization performs a little better than UID, FHP and FHP$_i$, while IS

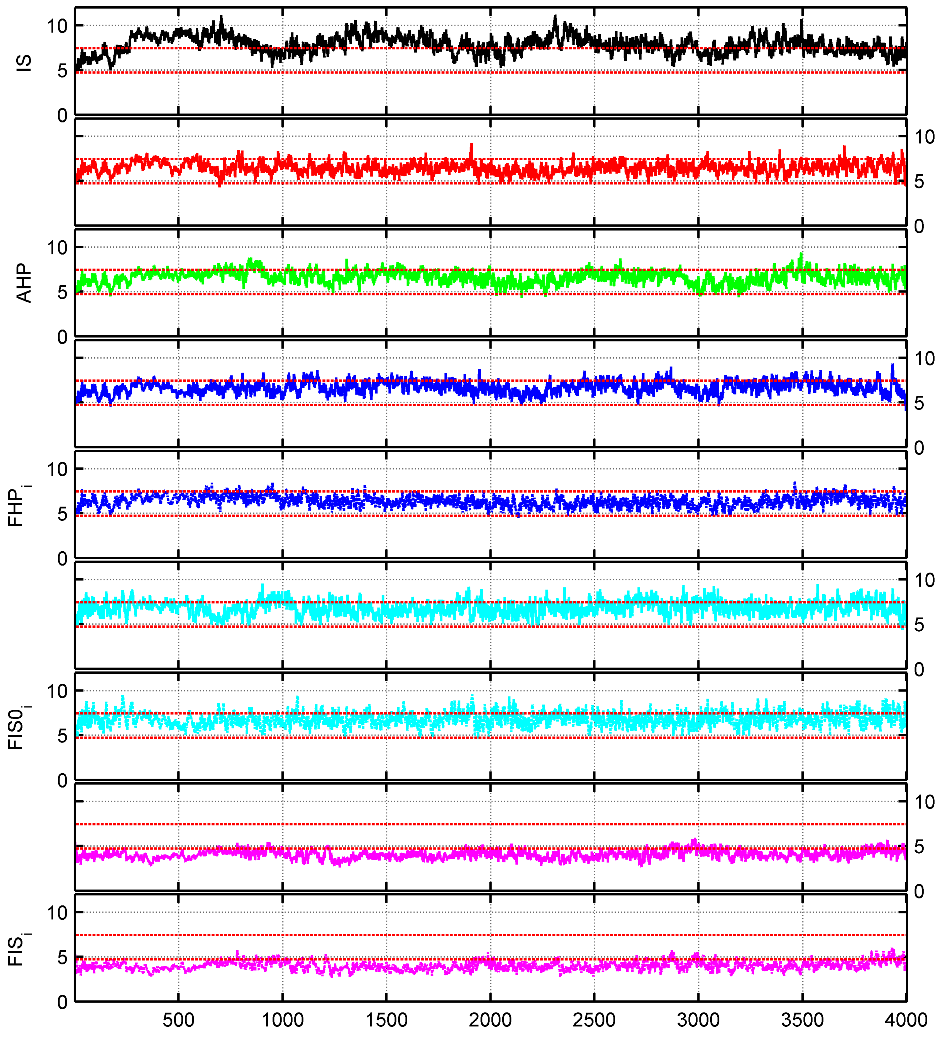


Figure 9.14: *Average NEES values in Experiment 12.b of the stereo setup, with the exact initialization of the viewing ray. In this experiment IS performance is comparable to other parameterizations. Dashed red lines are the limits of the* 95% *confidence interval. All parameterizations ensure a good level of consistency: average NEES value is, for the most of the time, between the bounds of the confidence interval. FIS and* FIS_i *parameterizations tend to be conservative, i.e., they are close to the lower bound of the confidence interval.*

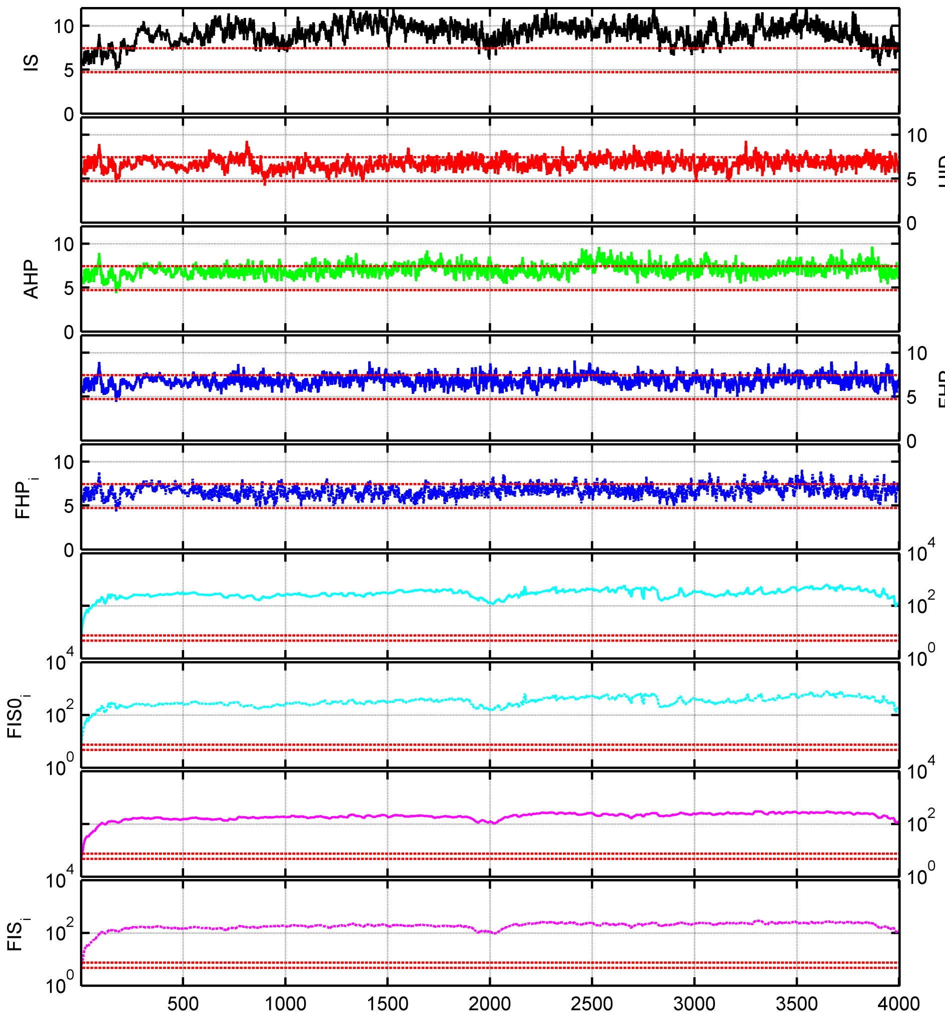


Figure 9.15: *Average NEES values in Experiment 12.b of the stereo setup, with the noised initialization of the viewing ray. Comparison with Figure 9.14, where the exact initial of the viewing ray was used, shows that FIS parameterizations (in all its versions) loose the consistency, while average NEES values are slightly increased for others parameterizations and they tend to be optimistic. FHP_i shows the best performances, remaining in the consistent range for most of the time.*

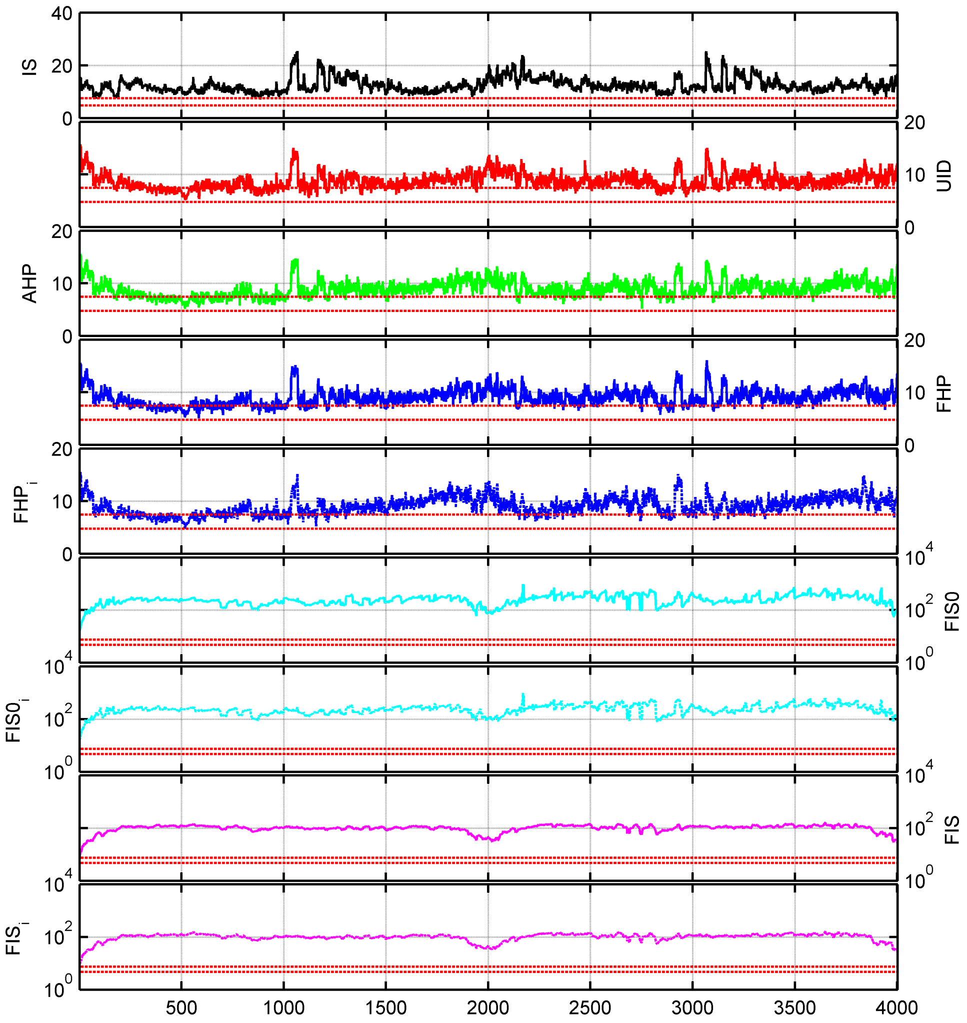


Figure 9.16: *Average NEES values in Experiment 12.b of the stereo setup, with the noised initialization of the viewing ray using the high resolution camera. Comparison with Figure 9.15, where the standard resolution camera was used, shows that all average NEES values are slightly increased for all parameterizations but FIS. FIS parameterization (in all its versions) performance is slightly improved by the use of the high resolution camera, although it remains far from the consistency range.*

Table 9.7: *Summary of results for the mean time in which the filter operates in the optimistic range in stereo experiments. Weighted average is performed on the percentage of operative time in the optimistic range from Tables 9.6, A.6 and A.8 from Appendix A with the experiment length (*800 *for Experiments 8.a to 11.c,* 4000 *for Experiments 12.a to 14.c). These results are graphically shown in Figure 9.17. In bold the parameterizations that reach the best results for the three different setup and the overall best. Notice that FIS parameterization, in all its versions, is not compared to other parameterizations, since it performs very well with the exact initialization and very bad with the noised initialization, due to its formulation.*

	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
Exact init.	92.6	46.5	**45.4**	46.8	50.3	44.6	43.2	0.2	0.2
Noised init.	97.5	64.9	63.3	**62.8**	63.0	100	100	100	100
Noised init. HD	99.9	95.8	**95.5**	95.7	**95.5**	100	100	100	100
Mean	96.7	69.1	**68.1**	68.4	69.6	81.5	81.1	66.7	66.7

and FIS perform bad. The analysis of total inconsistency shows that FIS0 inconsistency grow less than other parameterizations with the high camera resolution and it decreases when the FIS is used with the additional noise variance equal to 1.

9.1.2 Hybrid EKF SLAM Test with Loop Closure

In this section we want to show the problems that the frame anchored parameterizations (i.e., FHP and FIS) have when a loop closure step of the Conditional Independent SLAM framework is performed and we want to verify that the Hybrid Indirect EKF SLAM approach, with its parameterizations expressed in the indirect form, solves this problem. Moreover, we want to show that the point anchored parameterizations, such as UID, do not suffer of issues in loop closure, since problems are given by the use of quaternions.

Recalling briefly Section 5.6.1.6, when a loop is detected some landmarks that are already in a submap are seen from a different map. The landmarks involved in the loop have to be copied from the original map to the current map along all the intermediate submaps. This implies that, if FHP or FIS parameterizations are used, the anchor frames of the landmarks have to be transformed and expressed in the current map reference system. Doing this, the orientation of some anchor frames coming from different maps may results in a rotation which is far from the identity rotation. Consequently, the scalar element of the quaternion in the anchor frame results far from 1 and the approach proposed in Section 6.4.7.3, which is based on the representation of quaternions with the axis part only is not applicable. This does not happens with the hybrid approach proposed in Chapter 7, since rotations are always represented in a local form. The point anchored parameterizations (i.e., UID and AHP) do not suffer from this problem, since rotations are not used to represent landmarks, but they are used only in the current robot pose description.

We run a simulation in a cloister-like environment using the Conditional Independent SLAM approach. The cloister has a size of 22x22 meters on two different planes, respectively at the height $z = -0.5$ and $z = 0.5$ meters. The robot performs a circular trajectory

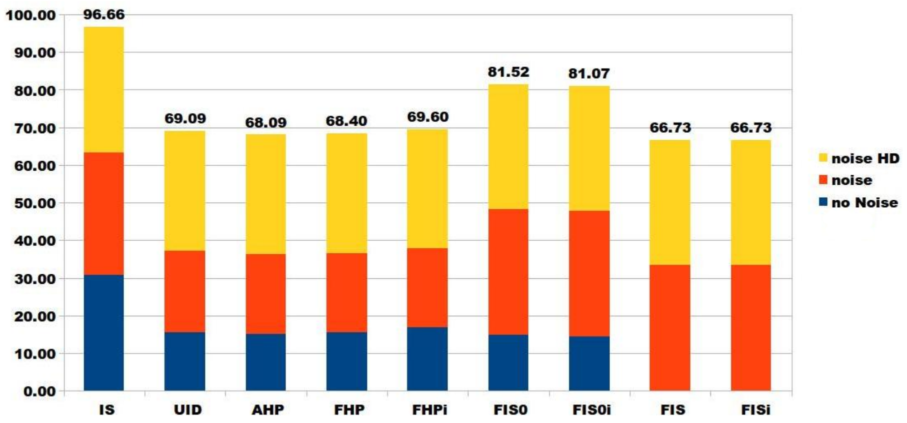


Figure 9.17: *Graphical summary of results for the mean time in which the filter operates in the optimistic range of Table 9.7 in the three different setup (i.e., with exact initialization, with noised initialization and with noised initialization using the high resolution camera). The bar height corresponds to the mean value reported in the table; each bar is partitioned in the value obtained in the three different setups.*

Table 9.8: *Summary of results of the total of average inconsistency value in the stereo setup. The total inconsistency is calculated by summing the values of average inconsistency of Tables A.1, A.7 and A.9 from Appendix A. In bold the parameterizations that reach the best results for the three different setup and the overall best. Notice that FIS parameterization, in all its versions, is not compared to other parameterizations, since it performs very well with the exact initialization and very bad with the noised initialization, due to its formulation.*

	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
Exact init.	2164	45	**43**	44	44	20	20	9	9
Noised init.	2084	49	**46**	**46**	47	4478	4564	1965	1919
Noised init. HD	18717	477	**451**	**451**	465	4621	4620	1277	1299
Total	22964	571	**540**	541	556	9119	9205	3251	3227

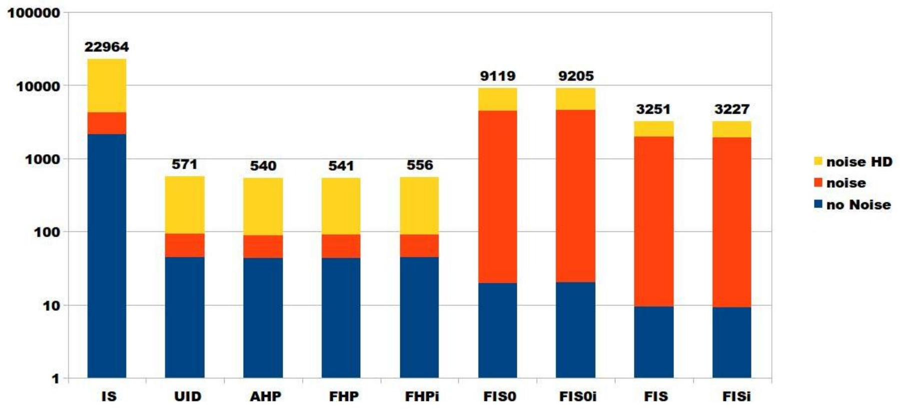
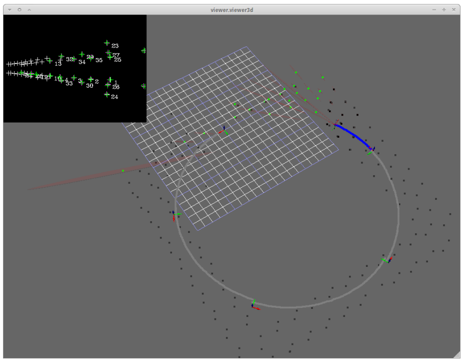


Figure 9.18: *Graphical summary of results of the total of average inconsistency value Table 9.8 in the three different setup (i.e., with exact initialization, with noised initialization and with noised initialization using the high resolution camera). The bar height corresponds to the mean value reported in the table; each bar is partitioned in the value of the three different setup, but it has to be noticed that the logarithmic axis on the ordinate prevents comparisons between values of parameterizations that perform significantly different.*

of 20 meter diameter on the plane $z = 0$ and the (monocular) camera points forward in the direction of the motion. A complete turn of the cloister is completed in 800 steps. The SLAM system does not use odometric information, thus it uses a constant velocity motion model. The submap system performs a map transition when the number of landmarks in the map is greater than 50 and generates 6 different maps. A loop closure is triggered at step 700, when some landmarks of the first map are seen in the sixth map. We stop the execution at step 850, i.e., after the completion of a loop of the cloister. The simulation is performed with the UID parameterization, the FHP parameterization and the FHP$_i$ parameterization, i.e., with the indirect version of the FHP parameterization used in the Hybrid Indirect EKF SLAM system.

Let analyze first the behavior of the FHP parameterization to show the problems it has in the back propagation step. First of all, we propose a visual inspection of what happens when a loop closure is performed with the FHP parameterization. Figure 9.19(a) shows the estimated robot trajectory up to the last closed submap with a gray line, the trajectory estimated in the current map with the blue line and the landmarks, represented by the black and green points just before the loop is detected. After the loop is closed the robot pose and the map suffer an erroneous correction, thus they disappear from view, as shown in Figure 9.19(b). The analysis of the covariance matrices of the shared element between submaps after the loop closure is performed, i.e., after the landmarks involved in the loop closure step have been copied through the submaps, shows that the presence of quaternions that represent big rotations causes the ill conditioning of the matrix, since the representation of rotation with the only axis part is not adequate to this case. When FHP parametrization is used the condition number[3] assumes values greater than 10^{19}, while

[3]Refer to [11] for details on linear algerbra and condition number.


(a) Before Loop Closure

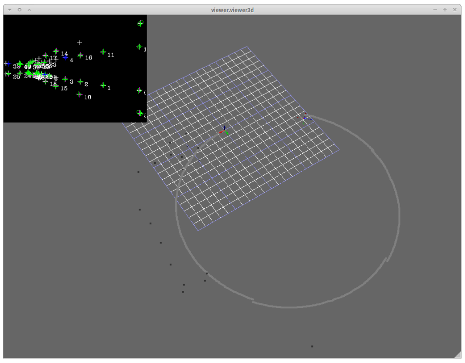

(b) After Loop Closure

Figure 9.19: *Two screenshots from the execution of a SLAM problem in a simulated cloister with the FHP parameterization. The Conditional Independent SLAM system is used and a loop closure is triggered when the robot revises landmarks that are already in the map. Figures show the estimated robot trajectory up to the last closed submap with a gray line and the trajectory estimated in the current map with a blue line. The landmarks are represented by colored points: black points represent landmarks in the map that are not currently visible while green points represent landmarks in the map that are currently visible. The reference system of each submap is indicated by a Cartesian reference system, where x, y and z axis are represented respectively by red, green and blue arrows. (a) is taken just before the loop is detected, while Figure (b) is taken after the loop closure. In (b) the current robot pose and the landmarks are disappeared, since the estimate suffers an erroneous correction due to the presence of an ill conditioned covariance matrix.*

with UID parameterization and FHP$_i$ parameterization it is lower that 10^9. This confirms that the FHP parameterization generates an ill conditioned problem, while the use of the Hybrid Indirect EKF SLAM system allows the use of framed parameterizations such that FHP$_i$.

UID and FHP$_i$ parameterizations, respectively applied to the direct and Hybrid Indirect SLAM formulation, are able to treat the loop closure problem and we use this example to compare their behavior in the estimation process. We analyze the *smoothed trajectories*, i.e., the trajectories reconstructed by the sequence of anchor points or frames stored in the shared part of the landmarks after the entire estimation has been performed. Consequently, we do not have an estimation of the robot pose at each time step, but only some *milestones* or *keyframes* of the trajectory, corresponding to the time step in which some new landmark has been added to the map. It has to be noticed that using UID we can reconstruct only the past robot positions, which are stored in the anchor points of the landmarks, while FHP$_i$ allows to reconstruct the complete past robot poses, thanks to the presence of anchor frames. We compare only the past robot positions, while the orientation information that are available with the FHP$_i$ parameterization are not evaluated. The trajectory is said to be *smoothed* since past robot poses are refined using also successive measurements performed in the system.

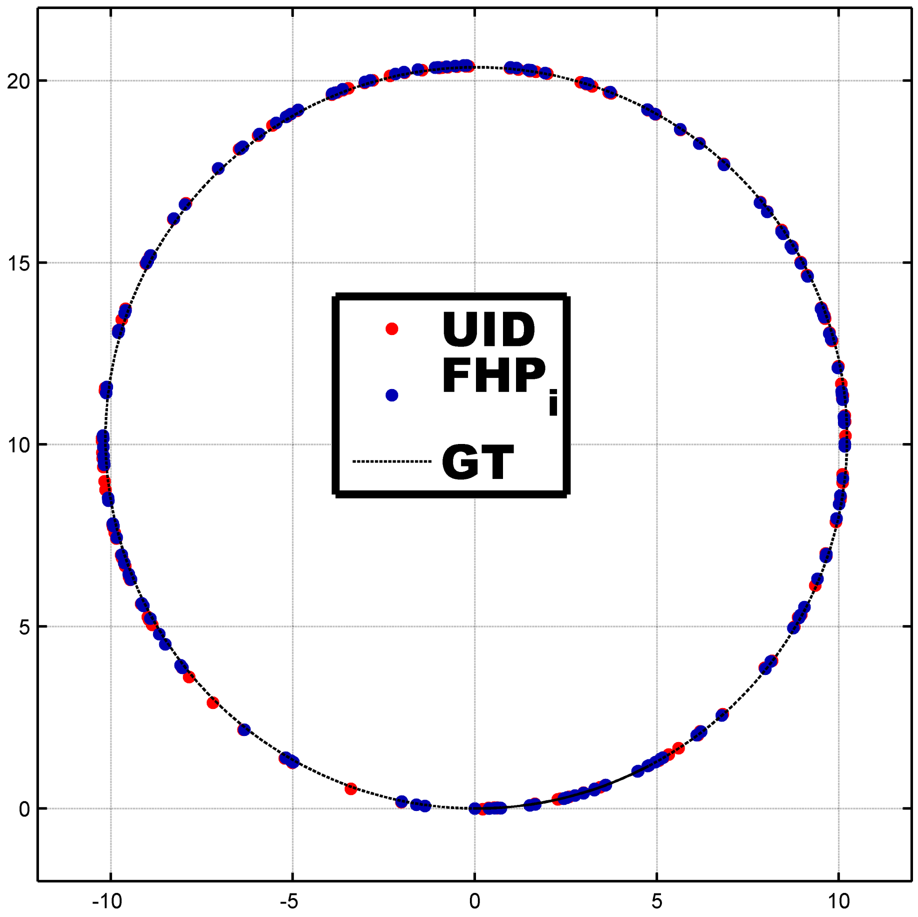


Figure 9.20: *Top view of the estimated smoothed trajectory for the UID (red points) and FHP$_i$ (blue points) parameterizations compared with the nominal trajectory, i.e., the ground truth (GT) in black.*

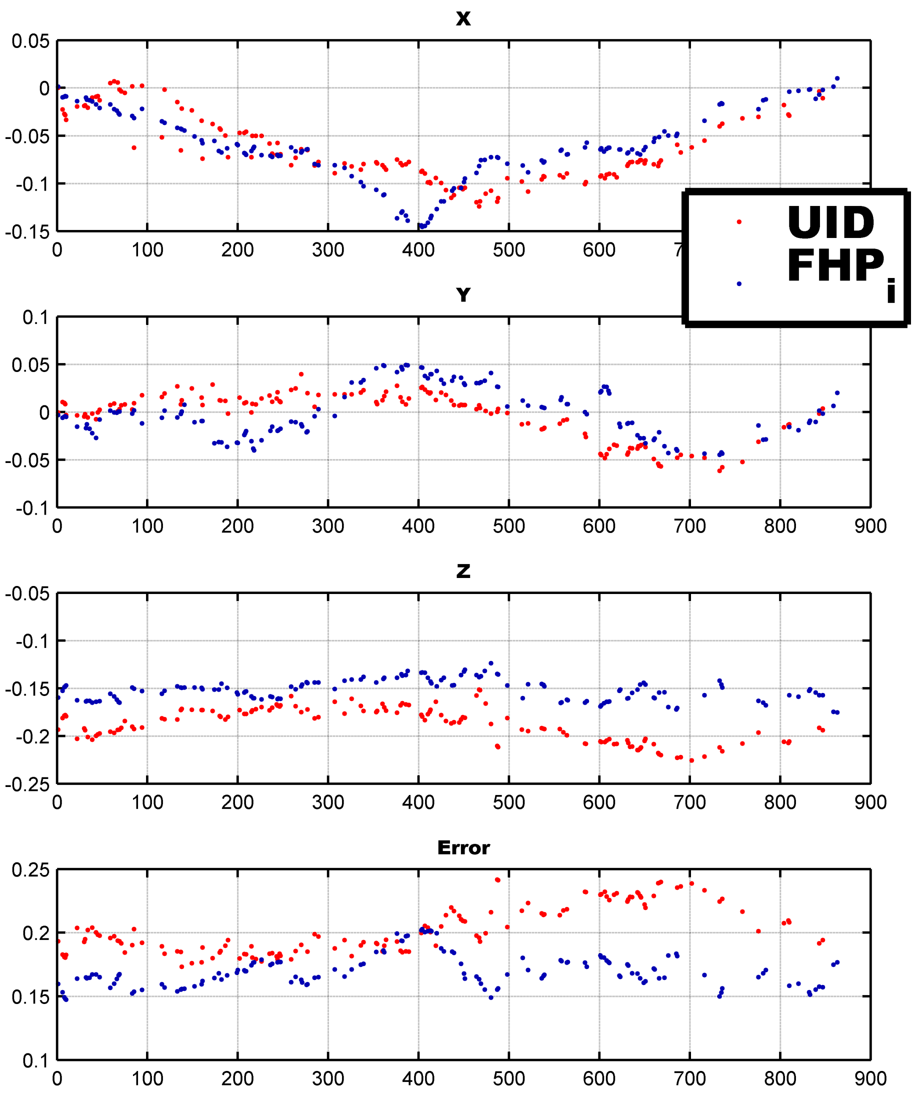


Figure 9.21: *Errors committed in the estimation by the two parameterizations on x, y and z component and the Euclidean distance from the exact position.*

Since the experiment we run uses a single camera and the odometry is not used in the motion model, the trajectory and the map result reconstructed up to an unknown scale factor. Consequently, to compare the solutions, reconstructed trajectories have been scaled to the nominal 20 meters diameter circular trajectory. Figure 9.20 shows the top view of the estimated smoothed trajectory for the UID and FHP$_i$ parameterization compared with the nominal trajectory (i.e., the ground truth). From this view it is not possible to distinguish if a parameterization performs better or worse than the other, since both the trajectories are accurately reconstructed and the loop was properly closed. Figure 9.21 shows the errors committed in the estimation by the two parameterizations on each component of the position and the Euclidean distance from the exact position. From this analysis can be concluded that the two approaches perform substantially equal, although there is a small advantage, i.e., a lower error in the estimation, for the FHP$_i$ parameterization. This confirms what was emerged from the parameterization evaluation presented in Section 9.1.1.

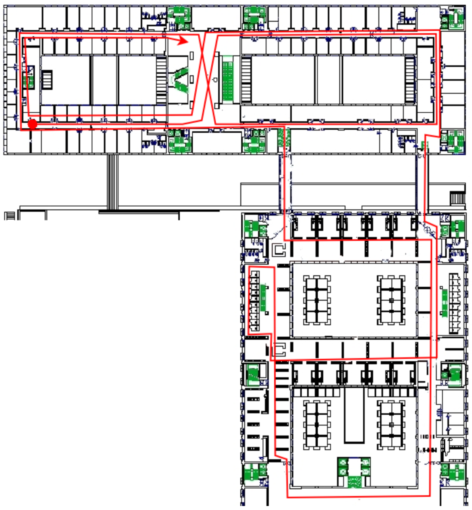
9.1.3 Evaluation On Real Data

We tested the proposed parametrizations also in a real setup, always using the MoonSLAM framework. As for the simulated experiments, we use the odometry of the robot as input in the motion model of the system. We have tested both the monocular and the stereo setup using almost[4] the entire indoor dataset *Bicocca_2009-02-27a* from the RAWSEEDS project [9], covering a path of about one kilometer. The architectural map of the building in which the dataset was collected and the path covered by the robot is sketched in Figure 9.22(a) and the robot, equipped with various sensors, is shown in Figure 9.22(b) The monocular camera has a resolution of 320×240 pixels with a focal length of about 200 pixels, it captures images at 30Hz and we used about 57000 frames (32 minutes). Each camera of the stereo dataset has a resolution of 640×480 pixels, with a focal length of about 660 pixels and a frame rate of 15fps; the baseline of the rig is about 18cm and we consider the same path as for the monocular case which totals about 28500 frames. Figure 9.23 shows a screenshot which has been taken during the execution of a monocular run.

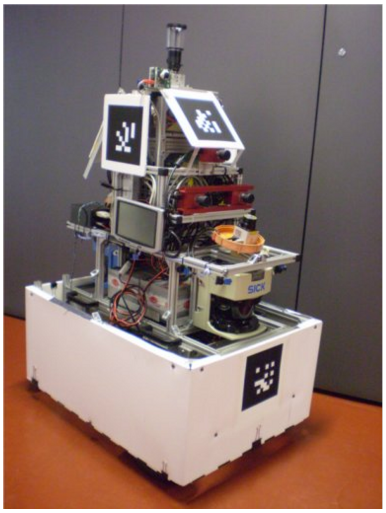
Both monocular and stereo algorithms perform active-search-based SLAM as for the simulated experiments. For the monocular case we initialize at most 16 landmarks in the first frame and search for new landmarks when less than 50 landmarks are predicted in the current image, with a limit of minimum two landmarks added each time. For the stereo case we initialize at most 25 landmarks in the first frame and search for new landmarks when less than 25 landmarks are predicted in the current image, without limits in addition. The difference in the initialization policies are due to an empirical tuning of the system. In the stereo case, landmarks are always initialized in the left camera reference frame, while measurement are obviously performed on both cameras.

Each landmark is described by a patch of 11x11 pixels area around a salient point, which is detected by FAST [83], while matching is done by normalized cross-correlation. Initial depth is set to 10 meters (corresponding to 0.1m^{-1} of inverse depth) with uncertainty on the inverse depth of 0.5m^{-1} both for monocular and stereo case. The update step is performed using the 1-Point RANSAC technique [16], which was briefly introduced in

[4]We limit the analysis of the dataset on the part where the *extended ground truth* (GT), i.e., a reference for the real trajectory, is available. For this dataset the GT covers the initial 80% of the dataset.



(a) Architectural map



(b) Robocom robot

Figure 9.22: *(a) The architectural map of the building in which the dataset was collected and the path covered by the robot. (b) The robot used to collect RAWSEEDS datasets. It is equipped with four laser scanner, a sonar belt, a monocular camera, a trinocular camera and an omnidirectional camera.*

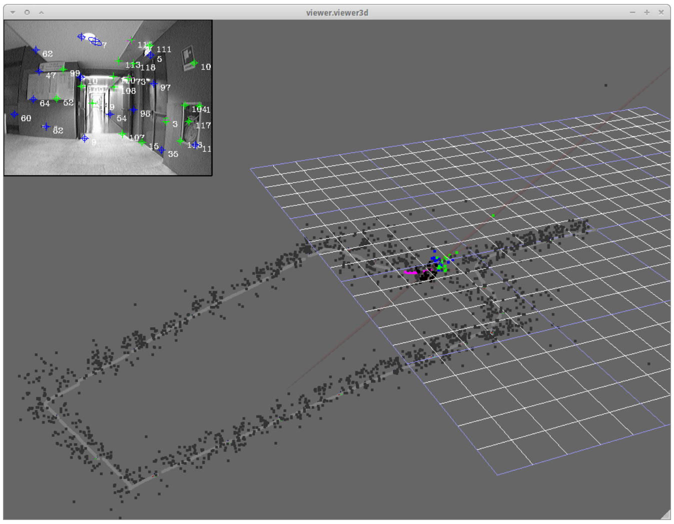


Figure 9.23: *A screenshot of the MoonSLAM framework performing monocular SLAM on a real dataset.*

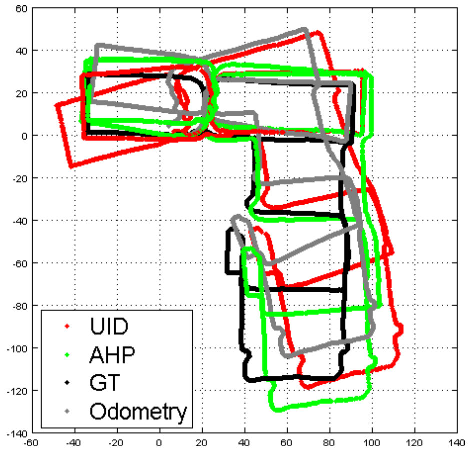
Section 5.4.6.

The submapping technique is applied with a maximum map size of 70 landmarks. We have to underline that the presented implementation does not perform the *loop closure* procedure of the CISLAM system, being not implemented in our framework the *loop detection* procedure. It has to be noticed that loop closure could reset the errors cumulated along the path before the loop closure point; in our case the cumulation of errors continues and this provides and idea of the accuracy obtained by the different parameterizations.

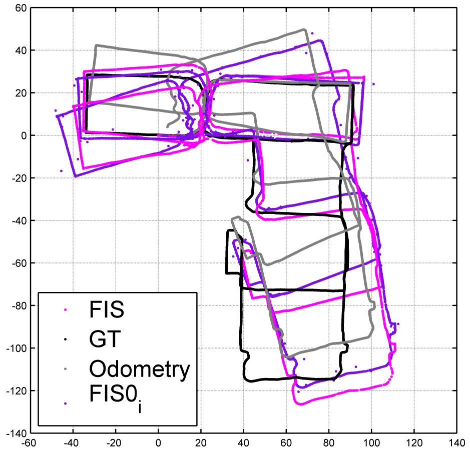
Figures 9.24 and 9.25 summarize the results respectively of the Monocular and Stereo SLAM for the parametrizations UID, AHP, FHP, FHP$_i$, FIS0 and FIS0$_i$ while IS is not treated in this run due to its poor performance. The *extended ground truth* (GT), i.e., a reference on the real trajectory available for the RAWSEEDS datasets, is plotted to allow comparison together with the robot odometry. Figures show the top view of the *smoothed trajectory*, i.e., of the trajectory reconstructed by the sequence of anchor points or frames stored in the shared part of the landmarks taking advantage of the back-propagation step in the Conditional Indipendent Submapping. Figures 9.26(a) and 9.26(b) show the top view of the map points (with crosses) on top of the *smoothed trajectory* for the FHP$_i$ parametrization respectively for the monocular and the stereo case.

From these experiments we see how all parametrizations show quite good performance in the reconstruction of the trajectory and the map. Since we used the odometry in the motion model for the monocular case we do not observe any significant *scale drift* phenomenon, as already observed in [15]. It looks like that the use of the stereo setup does not produce improvements with respect to the monocular setup, but we have to consider that the real dataset presents several possible sources of errors that may influence the final results. Thus, we do not concentrate our attention on the comparison between monocular and stereo systems.

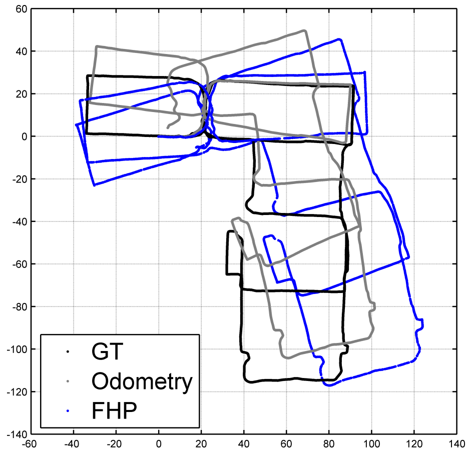
It is difficult from these results to state which parametrization is the best, if any; when using real data, there are several sources of errors, which are not properly modelled in the state of the art approaches, so it might be difficult to discriminate whether the performance gain (or loss) is fully due to the parametrization or other error sources. Different setups (e.g., changes in standard deviation of the noise considered in the model, different depth initialization, etc.) could also affect the results. On the other hand, a qualitative conclusion that can be safely drawn, is that the FIS0 parametrization and its indirect corresponding FIS0$_i$ parametrization shows a good performance also in a real setup, although they can not re-estimate the direction of the initial viewing ray.



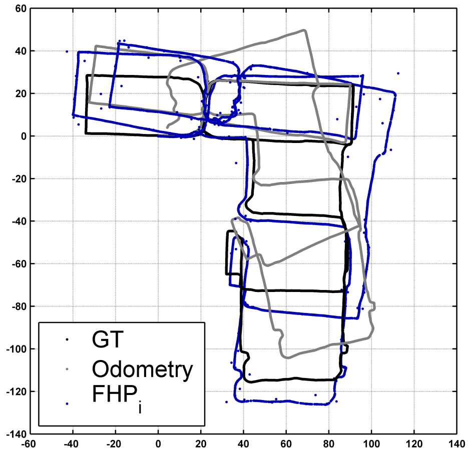
(a) UID and AHP



(b) FIS0 and FIS0$_i$

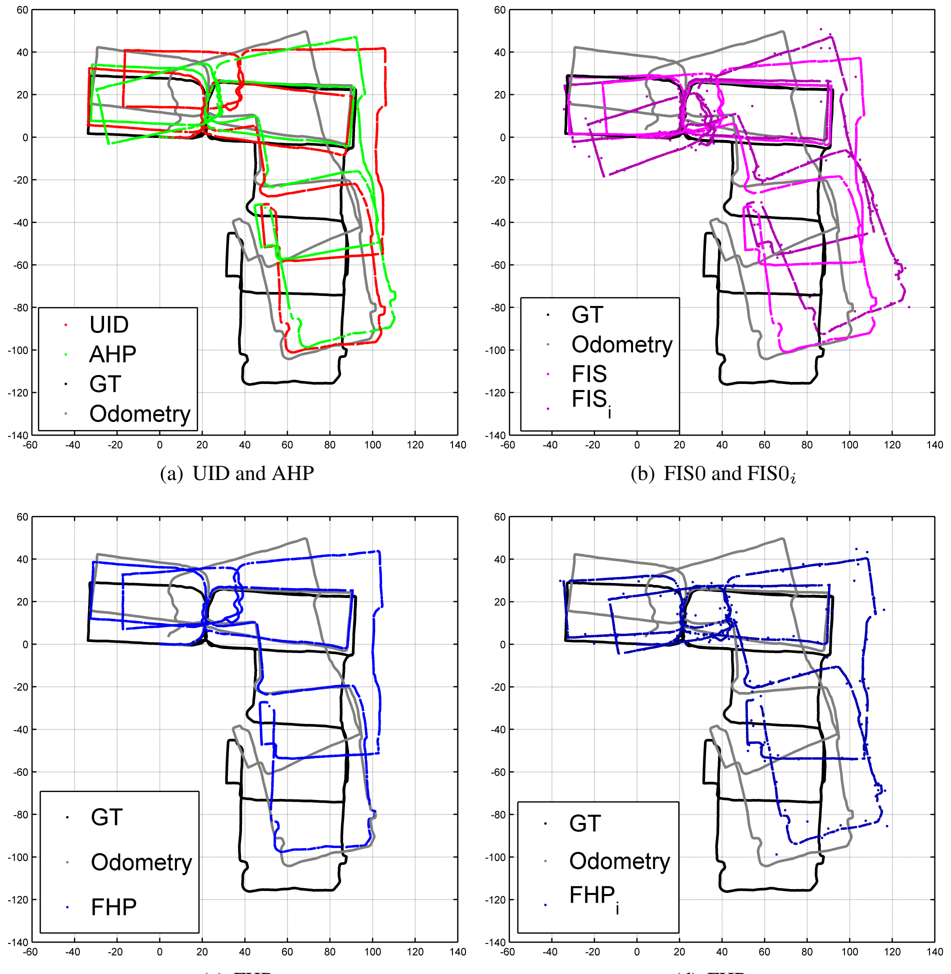


(c) FHP



(d) FHP$_i$

Figure 9.24: *Robot smoothed trajectory estimates on the real data for the Monocular setup. The trajectory is reconstructed by the positions stored in the anchor frames/-points from the EKF state.*



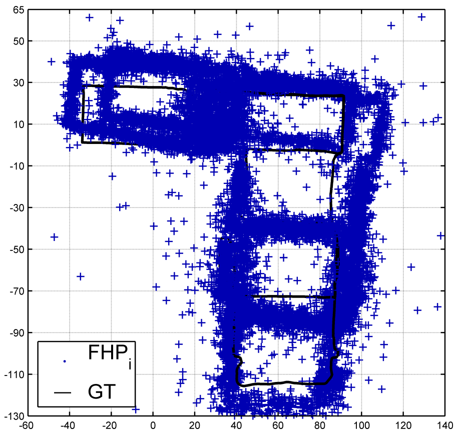
(a) UID and AHP

(b) FIS0 and FIS0$_i$

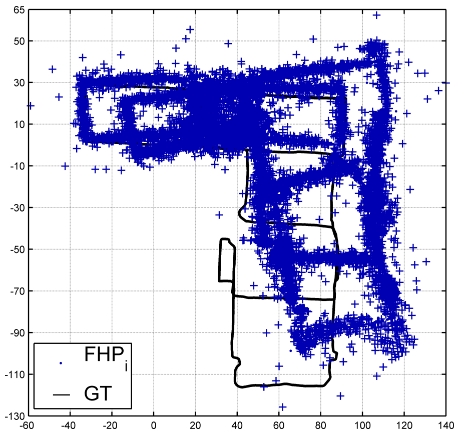
(c) FHP

(d) FHP$_i$

Figure 9.25: *Robot smoothed trajectory estimates on the real data in the stereo setup. The trajectory is reconstructed by the positions stored in the anchor frames/points from the EKF state.*



(a) Monocular



(b) Stereo

Figure 9.26: *(a) Robot trajectory and map estimate on the real data in the Monocular setup with the FHP_i parameterization. (b) Robot trajectory and map estimate on the real data in the Stereo setup with the FHP_i parameterization.*

9.2 CIBA Slam Evaluation

In this section we propose the evaluation of the proposed CIBA SLAM system both in simulated and real experiments. In particular, we test the CIBA SLAM system:

- in a simulated cloister-like environment,
- in a *annotated* real dataset, i.e., in a dataset where the data association is given,
- in a real dataset from the Rawseeds project.

In all these cases, results are compared with estimation performed without the Bundle Adjustement optimization of submaps.

9.2.1 Simulated Scenario

To test the proposed CIBA SLAM system we run a simulated experiment on a cloister similar to the one used for experiments of Section 9.1.1, but we perform here a pure monocular SLAM, i.e., we do not use the odometric data as input in the motion model. The simulated trajectory covers a planar circle with 10 meters radius in 800 frames. Random noise is added to the measurements and loop closure is not applied, thus the estimated trajectory accumulates errors even when a complete turn is performed. We compare the trajectories estimated by the CI SLAM and the CIBA SLAM system, i.e., respectively without and with the Bundle Adjustment on the submaps. Being in a pure monocular contest, we rescale the two solutions to the real trajectory to obtain a fair comparison between them.

The back propagation step is applied at each map transition and 4 submaps are generated in this specific experiment. Figure 9.27 reports both the on-line estimated trajectory

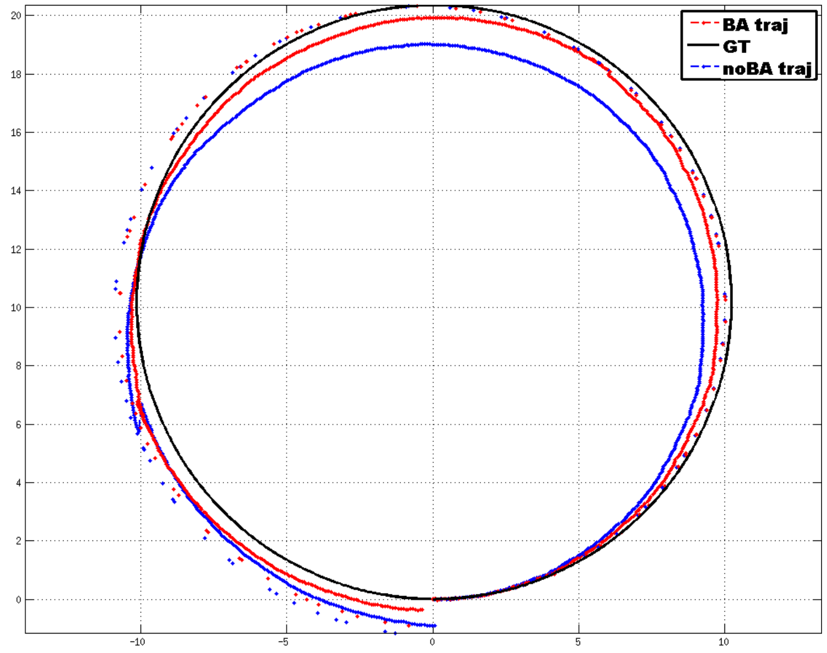
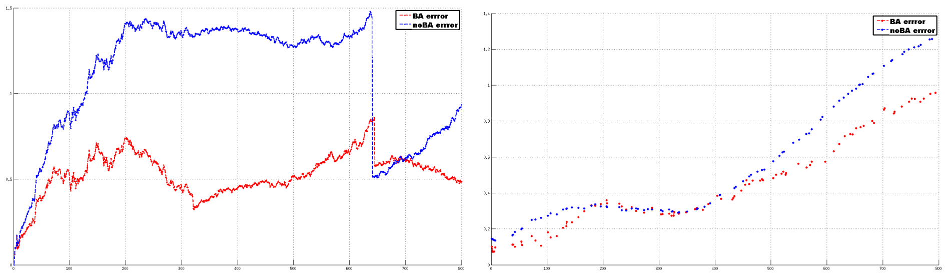
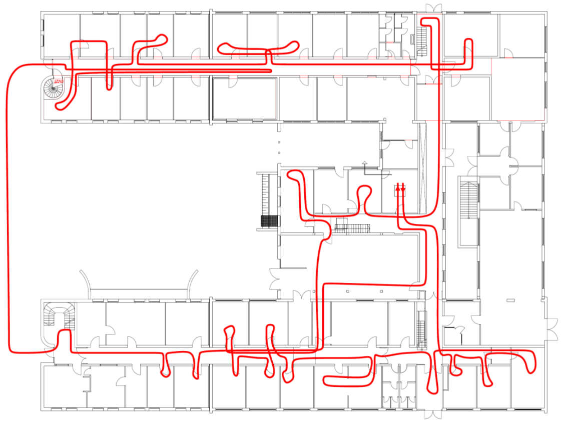


Figure 9.27: *Robot trajectory estimates on the simulated scenario with CI SLAM and with CIBA SLAM. Solid lines represent the trajectories estimated online, while the dotted are the trajectories extracted from the anchor frames that are in the filters at the end of the estimation after last back propagation step. Discontinuities in the online trajectories are due to the back propagation, which updates the last robot pose of each map, corresponding to the starting frame of the successive map.*

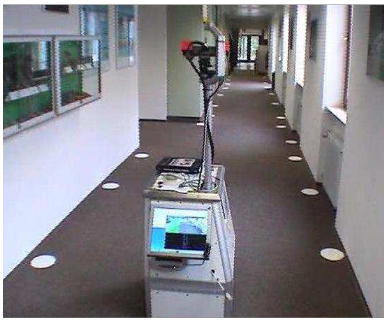


(a) On Line Trajectory Error (b) Smoothed Trajectory Error

Figure 9.28: *Error between the real robot pose and and its estimate, (a) for the on line trajectory, (b) for the smoothed trajectory. Discontinuities in the online trajectories are due to the back propagation, which updates the last robot pose of each map, corresponding to the starting frame of the successive map.*



(a) Architectural map



(b) The robot and the landmarks

Figure 9.29: *(a) Sketch of the path the robot traveled through the DLR building. (b) The robot used to collect the dataset and the artificial circular disks on the floor.*

(with solid line) and the smoothed trajectory, i.e., the trajectory reconstructed by the pose of the camera maintained in the submaps as anchor frames at the end of the experiment. The on-line trajectory obtained with the bundle adjustment of submaps results closer to the real trajectory, and this is valid for the smoothed trajectory too. This can be observed also in Figure 9.28(a) and Figure 9.28(b) where the error between the real trajectory and the estimated one, respectively for the on-line estimation and for the smoothed trajectory is reported. Notice that this experiment is not comparable with the one proposed in Section 9.1.2: here we use different value for the covariance of the noise to "stress" the system and to test it in a harder context.

9.2.2 Annotated Dataset

In this experiment we use the dataset presented in [50]. The dataset was recorded at the DLR (Deutsches Zentrum für Luft und Raumfahrt) Institute of Robotics and Mechatronics building using a mobile robot controlled by hand. The building covers a region of 60×45 meters and the robot path consists of three large loops within the building (plus a small outside path) with a total length of 505 meters discretized in 3297 steps. On the way the robot visits 29 rooms, as shown in the architectural map in Figure 9.29(a). The dataset provides the 2D odometric information, i.e., the relative movements on the x and y axis and the angular displacement around the z-axis. The odometry is accompanied with the 3×3 associated covariance matrix. Circular disks were outlaid on the floor throughout the building. The frames grabbed from a monocular camera, which had been intrinsically and extrinsically calibrated with respect to the robot reference frame, were processed with an artificial vision algorithm which detects the landmarks and transforms them with the underlying camera model to pure geometric measurements in robot coordinates. In each frame, each visible circular disk has a x, y-coordinate describing the position of the landmarks with respect to the robot reference system and a corresponding covariance matrix describing the measurement error. The dataset is annotated, i.e., each landmark has a unique label (i.e., an identifier) manually identified to provide data association ground truth.

The authors of this work aimed at collect a dataset which could be used by researchers to test and compare their algorithms in feature-based SLAM and in particular data association. We use here this dataset in a different way: since data association is given, we use the ground truth of the data association to feed our SLAM algorithms. In particular, since the dataset is provided in 2D, we have simply converted the odometric information in 3D, by adding null movement on the z axis and null rotation around x and y axis, and by creating a 6×6 covariance matrix with small variance on the diagonal of the axis which are not subject to movement. The landmarks measurements, which are provided as 2D points in robot reference frame, have been converted in image points thanks to the camera calibration parameters. The evaluation of the Jacobians of the projection function has been performed to propagate (linearly) the covariance associated with the 2D measures in image coordinates. By doing this we obtain a dataset for a 6 degree of freedom SLAM system.

We generate a reference ground truth on the trajectory and the landmark positions by using g^{2}o [49] on the 2D data. We create a graph with two types of nodes: the robot poses, each represented as 2D position and orientation, and the landmark positions, each represented as 2D points. Nodes are connected with two types of edges: consecutive robot poses are connected by an edge that contains the odometric information while each landmark is connected to the robot poses in which it has been observed by measurement edges. The solution of this problem with the Gauss-Newton algorithm results in the estimated trajectory and in the estimated map that we use as reference for the comparison.

We perform three different solutions with our system: the first is a monocular SLAM using UID parameterization, the second uses the Hybrid Indirect EKF SLAM system with FHP$_i$ parameterization and the third execution uses the FHP$_i$ parameterization and the Bundle Adjustement on the single submap, i.e., the complete CIBA SLAM system. Since the robot closes numerous small loops (e.g., it enters and exits from small rooms), we set the limit on the number of landmarks per map to 250 to allow the "automatic" detection of loops. This comes from the fact that landmarks that are in the active submap can be observed when the robot travels on an already explored area, while if landmarks are in different maps we need to call the loop closure procedure explicitly. We perform a single loop closure between submaps at frame 3000 (near the end of the dataset), when the robot closes a large loop after it had traveled the entire building.

Figure 9.30 shows the estimated *smoothed trajectory* (see Section 9.2.1 for details) for the three configurations and the reference ground truth coming from the 2D optimization. At a first glance, all the three solutions are equivalent, since their trajectories are almost overlapped. The analysis of the estimation error with respect to the ground truth, plotted in Figure 9.31, allows the analysis of the differences between the solutions. We can conclude that CIBA SLAM performs slightly better than the FHP$_i$ parameterization, although their errors are very similar. UID performs differently, but the maximum values of errors are comparable to FHP$_i$, although it presents some high peaks, especially at the end of the dataset.

It has to be considered that this experiment has been conducted on a very particu-

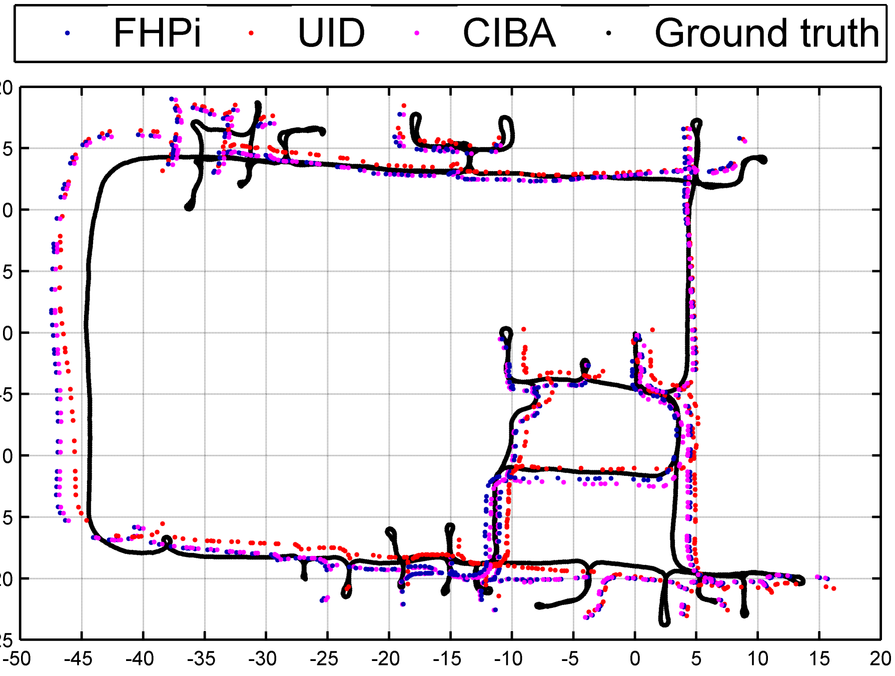


Figure 9.30: *Robot* smoothed trajectory *estimate on the annotated dataset of DLR building with UID parameterization, FHP$_i$ and the CIBA SLAM system with FHP$_i$ parameterization. A solution obtained by 2D optimization of the dataset is used as ground truth reference.*

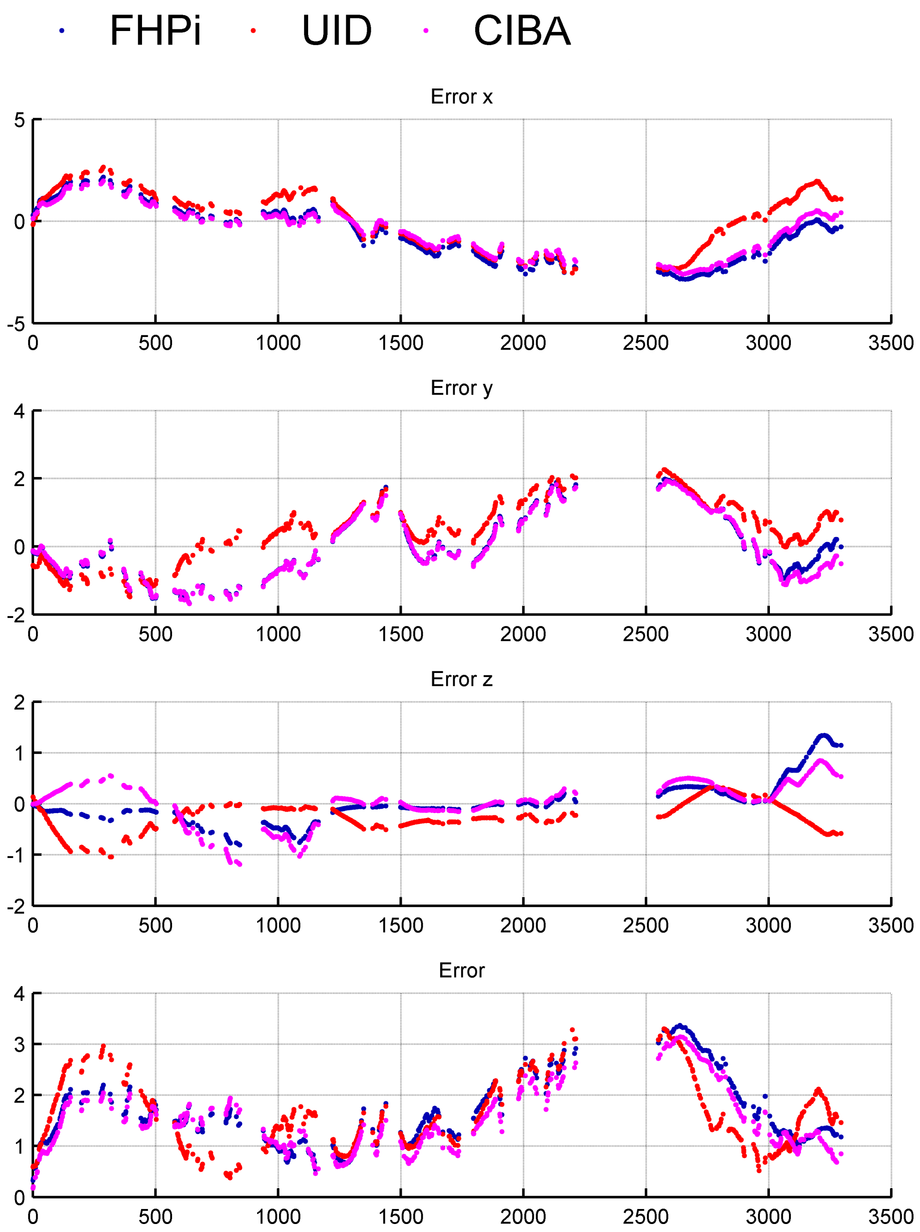


Figure 9.31: *Error between estimates on the annotated dataset of DLR building with UID parameterization, FHP$_i$ and the CIBA SLAM system with FHP$_i$ parameterization and the ground truth. Error is reported in the x, y and z components and as the Euclidean distance from the ground truth.*

lar dataset, which does not reflect the typical structure of the Visual SLAM application. In this dataset the robot has a rather large movement at each step, while the typical assumption in Visual SLAM is that the frame rate of the camera is sufficiently high to have small movements between frames. Moreover, in Visual SLAM, we assume to have about ten or twenty points visible per each image, while in this dataset the number of circular landmarks for image is very small. We conclude that, although it seems that the use of the CIBA SLAM system does not provide relevant improvements, it is not easy to draw definitive conclusions from this single experiment.

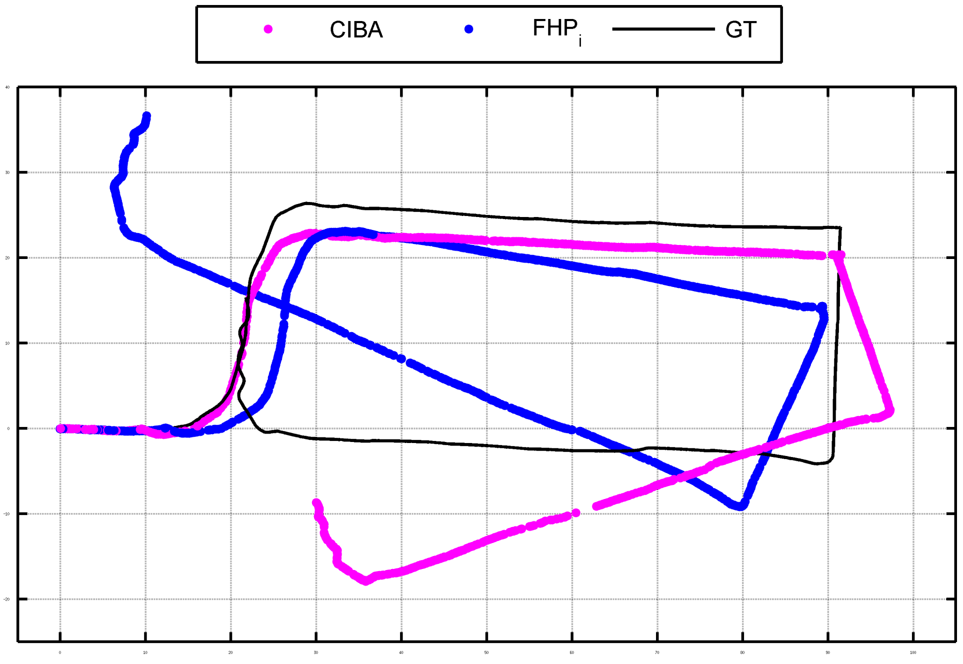
9.2.3 Real Experiments

We tested the proposed CIBA SLAM system also in a real setup, always using the MoonSLAM framework. In particular we use the same dataset of Section 9.1.3 but in this case we do not use the odometry of the robot as input in the motion model. We tested both a monocular and stereo setup with a constant velocity motion model for the prediction step of the filter. These can be considered the hardest case for a Visual SLAM system, in particular for the monocular case: a single camera does not offer information on the depth, the estimation of the motion without the aiding of the odometric information results particularly challenging. Moreover, in this setup the camera looks in the motion direction which is the worse case since few information on the parallax is generated. In particular, in the dataset, the robot performs in place rotations at the end of corridors while the camera is looking to a wall with a few number of features. We choose to test this case because it is quite common that cameras are mounted frontally on real robots, although this is not the best placement for the motion estimation.

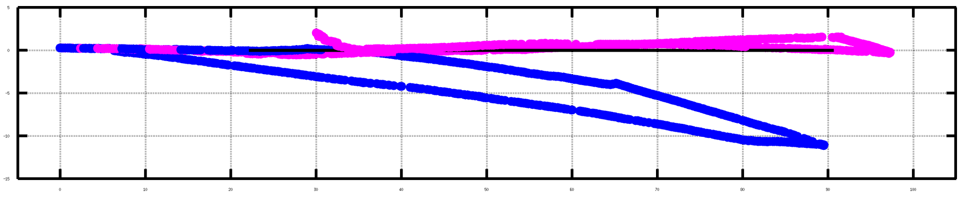
In the monocular setup we stop the execution after about 12000 frames, i.e., 400 seconds at 30Hz, when a complete loop of a building is completed. We consider not meaningful to continue the execution after this point, since the error accumulated is quite big and it can not be reset since we have not implemented a system for the loop detection when real data are used. In the stereo setup we stop the execution at the same point, corresponding to frame 6000, since the cameras in the stereo rig acquires image at 15Hz.

Figure 9.32 shows the trajectories estimated in the monocular setup with the FHP$_i$ parameterization with and without the bundle adjustment optimization of the single maps, i.e., with and without the CIBA SLAM system, compared with the ground truth available in the dataset. The trajectories have been rescaled to the ground truth, since they come from a monocular system and they are estimated up to a scale factor. The top views of the trajectories shown in Figure 9.32(a) are not very significant: both the trajectories are quite far from the ground truth reference. It can be noticed that big errors are accumulated at the end of corridors, where the robot turns in place and the rotation is not properly estimated. Figure 9.32(b) shows a side view of the trajectories. It can be noticed that in this case the effects of the CIBA SLAM system is evident: the trajectory estimated remains close to the $z = 0$ plane, while the trajectory estimated by the FHP$_i$ parameterization without the Bundle Adjustment step falls below the ground plane more than 10 meters.

Figure 9.33 shows the trajectories estimated with the stereo setup with the FHP$_i$ parameterization with and without the bundle adjustment optimization of the single map, i.e., with the CIBA SLAM system, compared with the ground truth available in the dataset. The top view of the trajectories shown in Figure 9.33(a) confirms the benefits coming from the application of bundle adjustment: the estimated trajectory is closest to the ground truth and

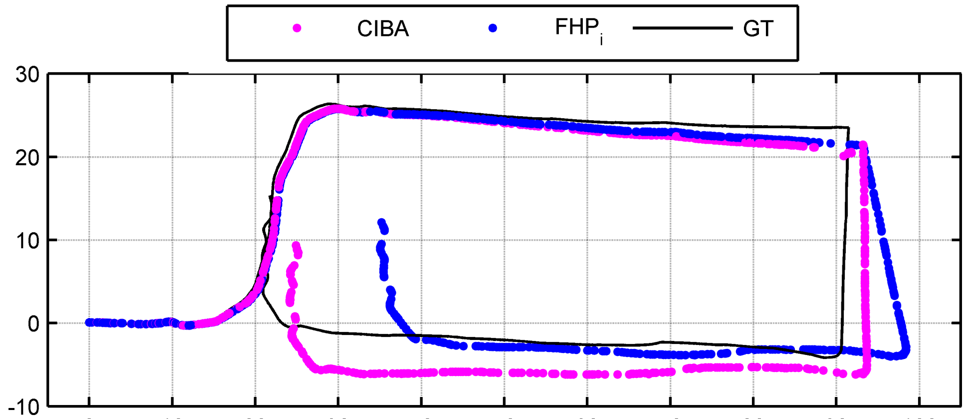


(a) Top view

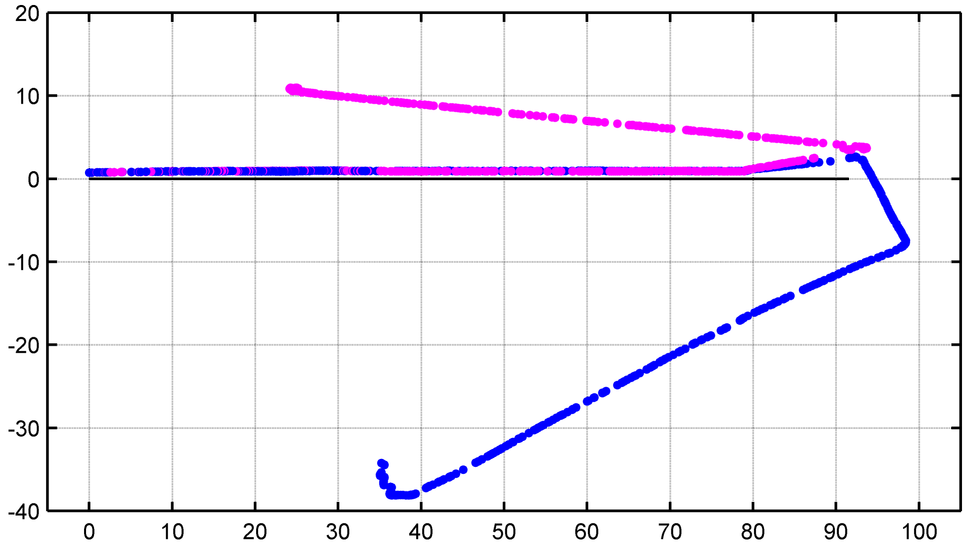


(b) Side view

Figure 9.32: *Estimation performed in a "pure" monocular setup, i.e., without the use of odometric information and with the same covariances on the velocity motion model for all the axes. The figures compare the trajectory estimated by the Hybrid Indirect EKF SLAM system using the FHP$_i$ parameterization with the CIBA SLAM system, which uses the same parameterization but performs Bundle Adjustment at each submap. (a) shows the top view of the trajectories, (b) shows a side view. The result shows how the Bundle Adjustement correction is useful: the estimation of the z coordinate, i.e., the vertical quote of the robot, which has to be zero, since the robot is moving on a plane, is properly estimated by the CIBA SLAM system, while it is estimated erroneously without it.*



(a) Top view



(b) Side view

Figure 9.33: *Estimation performed in a "pure" stereo setup, i.e., without the use of odometric information and with the same covariances on the velocity motion model for all the axes. The figures compare the trajectory estimated by the Hybrid Indirect EKF SLAM system using the FHP$_i$ parameterization with the CIBA SLAM system, which uses the same parameterization but performs the Bundle Adjustment at each submap. (a) shows the top view of the trajectories, (b) shows a side view. The latter result shows how the Bundle Adjustement correction is useful: the estimation of the z coordinate, i.e., the vertical quote of the robot, which has to be zero, since the robot is moving on a plane, is better estimated by the CIBA SLAM system, while it is estimated erroneously without it.*

in place rotation are estimated with better precision. The analysis of Figure 9.33(b), which reports a side view of the trajectories, shows that the use of bundle adjustment corrects the estimation of the quote of the robot, that remains more close to the $z = 0$ plane. It can be noticed that without the application of bundle adjustment, the attitude of the robot is poorly estimated in presence of in place rotations and this results in the wrong estimation of the trajectory, which moves away from plane $z = 0$.

These results show that, differently from what happened in the experiment of Section 9.2.2, the contribution of the CIBA SLAM system is not negligible when real datasets are used.

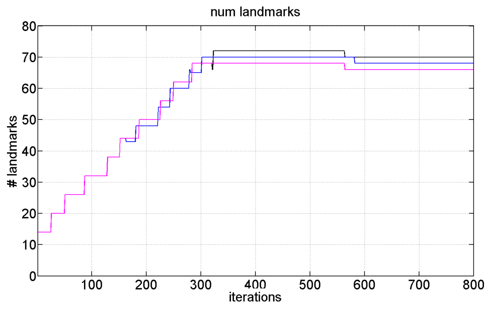
9.3 Time Performances Evaluation

The performance, in terms of execution time, of the EKF SLAM has been evaluated with both simulated and real experiments in order to validate whether we can obtain real-time performances on a standard PC. The experiments were conducted on a laptop equipped with an Intel Core2 Duo T9300 CPU running at 2.50GHz per core with 3.4GB of DDR2 RAM.

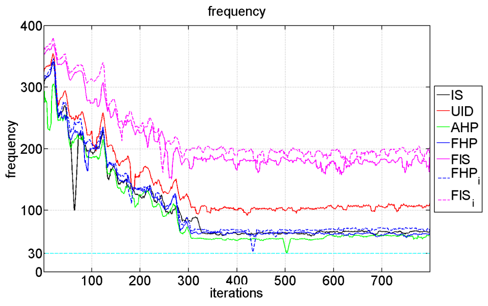
9.3.1 Simulated Experiments

Figure 9.34 shows the running frequency in one of the simulated experiments presented in Section 9.1.1.1, in particular Figure 9.34(b) shows the frequency for each parametrization at each iteration and Figure 9.34(a) shows the number of landmarks in the filter at each iteration. At iteration 400 the first turn of the cloister is completed; this implies that all the landmarks are already in the filter and the computation time becomes constant hereafter. Assuming a real-time nominal frequency of 30Hz for the camera, results show that all the parameterizations reach higher operative frequency also when the number of landmarks is about 70. The number of anchor frames/points generated during the execution is 11. In particular, the addition of 5 landmarks sharing the same anchor point/frame enhances FIS performance, especially in its indirect formulation, where the anchor frame dimension is reduced to 6 elements in the filter. AHP is the slowest parameterization, since it has the biggest landmark part, while FHP and FHP$_i$ perform a little better, similarly to IS. UID performance is in the middle, reaching the operative rate of about 100Hz. We should notice here that in simulated experiments the update step is conducted with an iterative procedure that negatively affects performance, since Jacobians are recomputed several times per iteration. Moreover, the iteration time is the cumulation of all the operations that are involved in the SLAM algorithm, including accessory operations such as the simulation of the nominal motion and the projection of the map points.

Figure 9.35 shows the performance in one of the simulated experiment from section-Section 9.1.1.2, where the stereo configuration is used. The slowdown is evident, but it can be noticed that the limit of 30Hz frequency is slightly exceeded only by the slowest parameterizations. Also in this configuration we have about 70 landmarks in the filter state with 11 anchor points/frames.

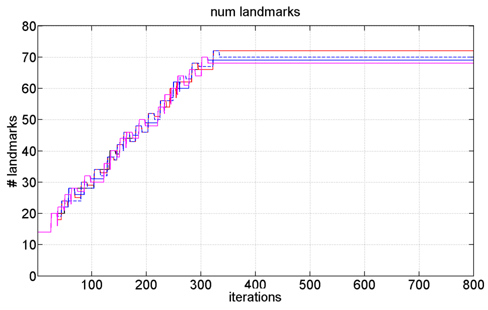


(a) Number of landmarks in the filter

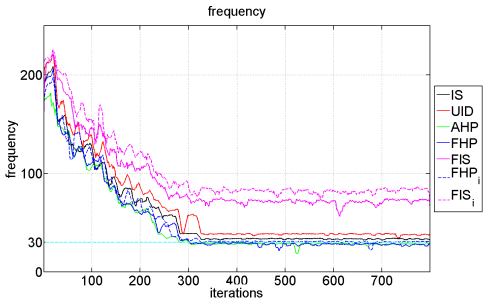


(b) Operative frequency

Figure 9.34: *Performance of EKF SLAM in simulated monocular experiments. In (a) the number of landmarks in the filter at each iterations; in (b) the operative frequency at each iteration; a cyan dashed line at 30Hz is provided as reference real-time performance.*

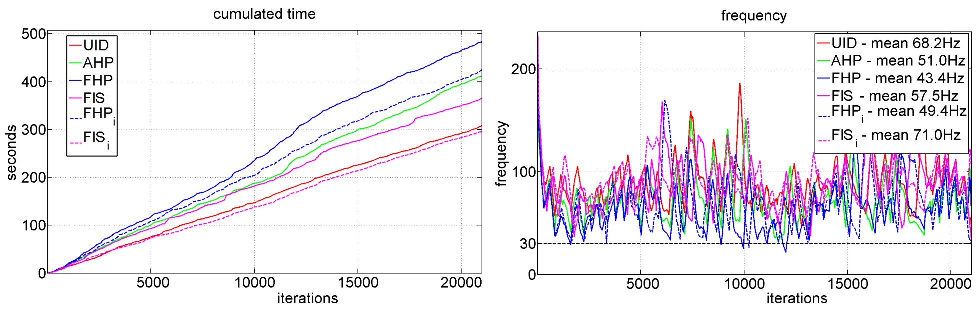


(a) Number of landmarks in the filter



(b) Operative frequency

Figure 9.35: *Performance of EKF SLAM in simulated stereo experiments. In (a) the number of landmarks in the filter at each iterations; in (b) the operative frequency at each iteration; a cyan dashed line at 30Hz is provided as reference real-time performance.*



(a) Cumulated time (b) Operative frequency

Figure 9.36: *Performance of EKF SLAM in the Monocular real data experiments. The use of the Conditional Independent Submapping framework allows a complexity of O(1) for each submap; this can be observed in (a), where the cumulated execution time is roughly linear in the number of iterations. (b) shows the operative frequency at each iteration and the legend reports the mean operative frequency for the different parameterizations.*

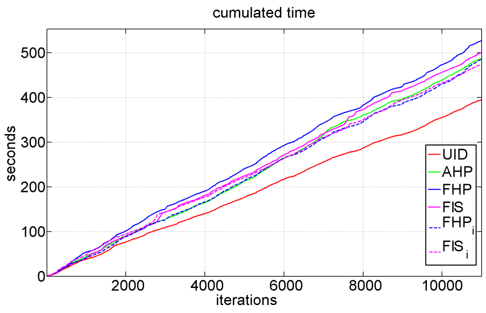
9.3.2 Real Experiments

Figure 9.36 shows the performance of all the parametrizations in the analysis of 21000 frames of a monocular dataset from Rawseeds[5] performed in Section 9.1.3. The MoonSLAM framework has been developed without any particular care to performance, thus no parallel thread execution is used and, in order to reduce overheads, the visualization is performed only one time every 30 frames. Moreover, the back propagation step of the CI SLAM framework is never performed here, since it can be implemented to be executed in parallel to the rest of the operations. Notice that in this case the update step is conducted with the 1 Point Ransac procedure, thus performance is not directly comparable to the simulated one. The map maximum size is incremented to 100 landmarks, since this limit is sufficient to guarantee real time performances.

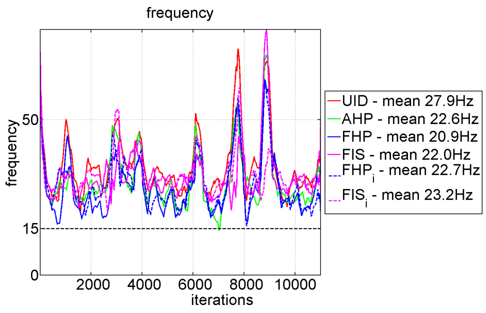
The use of the Conditional Independent Submapping framework [78] [77] allows a complexity of O(1) for each submap. This could be observed in Figure 9.36(a), where the cumulated execution time is roughly linear in the number of iterations. From Figure 9.36(b) we notice that all parametrizations, with a maximum number of 100 landmarks per map, allow the EKF based SLAM algorithm, in monocular case, to perform at a higher frequency than the camera frame rate (30fps). FIS$_i$ is the fastest parametrization (71Hz), followed by UID (68Hz), FIS (57.5Hz) and AHP (51Hz), FHP$_i$ (49Hz) and FHP (41Hz). From the simulated experiment we might expect better performance from FIS, but in this case it is slower than UID. This can be explained considering that in real cases some points that have been initialized may be lost and succesively deleted, reducing the advantage of the shared FIS parameterization.

Figure 9.37 shows the performance of all the parametrizations in the analysis of 11000

[5]The Rawseeds datasets are provided as a sequence of images (one image per file). This greatly slow down performances due to the high number of OS request to open and close files. We choose to evaluate time performances on uncompressed videos that were generated only on the initial part of the dataset.



(a) Cumulated time



(b) Operative frequency

Figure 9.37: *Performance of EKF SLAM in the Stereo real data experiments. The use of the Conditional Independent Submapping framework allows a complexity of O(1) for each submap; this can be observed in (a), where the cumulated execution time is roughly linear in the number of iterations. (b) shows the operative frequency at each iteration and the legend reports the mean operative frequency for the different parameterizations.*

frames of the stereo dataset from Rawseeds. In this case the camera frame rate is 15 fps and UID results the fastest parametrization (28Hz), followed by FIS, FIS$_i$, AHP and FHP$_i$ (more than 22Hz) and by FHP, that reaches the 20Hz. In this case we have no stressed the addition of a minimum number of landmarks, thus the advantages of the shared parameterizations are not enhanced and UID results the fastest parameterization. Considering that the stereo camera frame rate is 15Hz we can conclude that our system is still able to operate in real time in this setup.

9.3.3 CIBA Slam

We are interested in the evaluation of the performance of the system when the Bundle Adjustement of the submaps is used. We record execution time per step of the experiment presented in Section 9.2.1. Obviously, the addition of the bundle adjustment system impacts on the time performance, since at each map transition the map is optimimized, but it does not prevent the system from running in real time.

The addition of the bundle adjustment step, which has a constant cost per map, decreases the mean operating frequency from 53.7Hz to 45.9Hz but maintains the system in the real time operative range, although the operations performed at map transition, i.e., the bundle adjustment and the back propagation step, introduce a temporary overruns from the 30Hz limit frequency, as shown in Figure 9.38. We verify these considerations also with the real data experiments, concluding that the additional cost for the Bundle Adjustment step of each submap is tolerable and it does not prevent the system for running in real time.

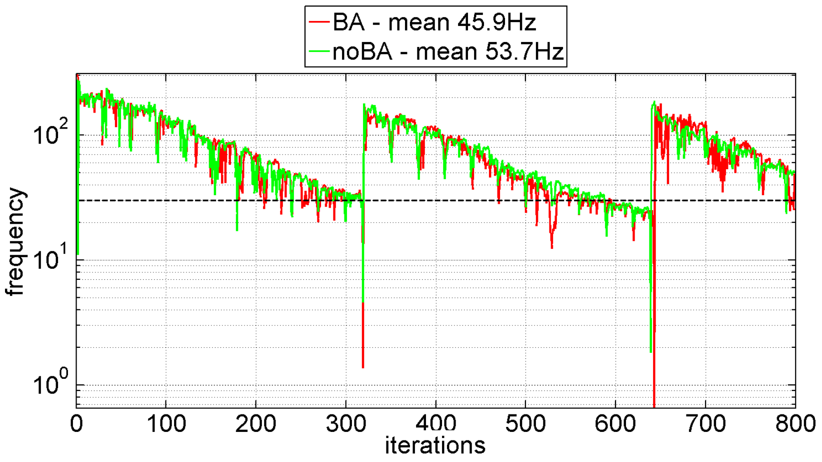


Figure 9.38: *Operative frequency of the CIBA SLAM system in the simulated experiment proposed in Section 9.2.1. The addition of the Bundle Adjustment step decreases the mean operating frequency from* 53.7*Hz to* 45.9*Hz. When map transition is performed the operative frequency falls down also when the Bundle Adjustment is not performed, since, for this experiment, the back propagation step is performed at each map transition. Notice the logarithmic scale on the* y *axis and a dashed black line that represents a reference at* 30*Hz for real time performances.*

CHAPTER 10

Conclusions and Future Works

In this chapter we draw some conclusion about this work and we introduce possible future activities and research directions.

10.1 CIBA SLAM System

In this work a modular approach to the solution of the multi camera SLAM problem is proposed following the next steps:

1. the parameterizations for monocular SLAM, have been deeply reviewed, extending what done in [90], [93] and [10] and two new parameterizations (FHP and FIS, see respectively Section 6.4.5 and Section 6.4.6) have been presented;
2. a multi camera large scale SLAM system is developed leveraging on the *Bi-Cam SLAM* approach proposed in [91] and [89] in conjunction with the the *Conditional Independent* submapping framework proposed in [78] (see Section 6.2);
3. the *feedback* and *feedforward* approach to *indirect EKF* have been mixed up in the proposed *Hybrid Indirect EKF-SLAM* setup (see Section 7.1.1) to solve issues related to the use of the CI SLAM framework with framed parameterizations; motion models (see Section 7.2) and the framed parameterizations (FHP and FIS) have been redefined in this formulation (see Section 7.3);
4. an additional module for Bundle Adjustment has been added to the CI-SLAM framework, performing a refinement of the maps, resulting in the *CIBA SLAM* system

(see Chapter 8). The optimization takes advantages of the particular structure of the Dynamic Bayesian Network (DBN) correspondent to the filter when the Hybrid Indirect EKF-SLAM with the framed parameterizations, creating a correspondent optimization graph which results are propagated to the filter state (i.e., to the Gaussian map) maintaining real time performance of the entire system.

The review of point parameterizations in the literature, in particular for UID [65] and AHP [90], has highlighted some concepts that have been usually leaved to the margins of the discussion. First of all we have highlighted the possibility to share the common anchor point, saving space in the state vector, as also observed in [43]. Moreover, sharing the common part of the state vector is mandatory for the use of the Conditional Independent SLAM framework, although this was not properly clarified, to the best of our knowledge, in the references works [78], [77] and [76]. The introduction of the so called *framed parameterizations* has enhanced the concept of anchor from the *anchor point* of UID and AHP to the *anchor frame* of *FHP* and *FIS*, introduced respectively in [10] and in this work[1]. Experimental results shows that there is no strong evidence to support any of those: they performs similarly in the consistency analysis and show similar results in real experiments. FIS parameterizations, although its assumption on exactness of the viewing ray introduces approximations, has shown the better performance in terms of operative frequency. Beside this, we have motivated the choice of the *framed* parameterizations (i.e., FHP and FIS) in order to develop the CIBA SLAM system, which takes advantage of the characteristics of these parameterizations to perform a non linear refinement of the maps, thanks to the storing of past camera poses (i.e., anchor frames) in the filter state.

We choose to develop a flexible and modular system, which can operate with a generic number of cameras. We take strong inspiration from the Bi-Cam SLAM approach proposed in [91] and [89] but we discard the implementation of the proposed auto calibration features of the algorithm to concentrate the attention on the SLAM system. This results in a modular system for visual SLAM that operates on a calibrated multi camera system. The system is built on the solution for the monocular case, since the addition of cameras corresponds simply to the addition of measurement equations without any particular structure or proper geometrical concepts (e.g., stereo SLAM is treated without any explicit reference to the epipolar geometry). The major advantage of this approach lies in the flexibility that it guarantees, since points may be observed in each camera independently from the others and consequently partially overlapped field of view or not overlapping field of view at all can be simply considered.

The use of the *indirect* form of the EKF allows storing in the filter only the estimate of errors between the true values and nominal values. Errors are local, thus close to zero, consequently rotations can be represented with a minimal representation, avoiding the use of quaternions in the filter. This has two main advantages: first, the number of elements needed to represent an anchor frame in the state vector is decreased to 6 instead of 7; second, the local representation of rotations avoids singularities in the covariance matrix. The analysis of the two possible approach to the indirect filtering, i.e., the *feedback* approach and the *feedforward* approach has driven the development of the *Hybrid Indirect* approach. The feedback approach has the advantage of resetting errors (i.e., integrate the state vector elements in the nominal values) at each iteration of the EKF, while in the feedforward

[1]As already observed, FHP and FIS parameterizations results very similar to the ones presented respectively in [43] and [74].

approach deltas may grow indefinitely. The delta reset procedure performed on the entire state vector has a complexity of $O(n^2)$, that may results in a too high computational cost. Considering that landmarks are static, we avoid to reset the deltas of landmarks, resulting in the *Hybrid Indirect EKF-SLAM*, that use a feedback approach for the variables in the state vector that represent the robot motion information, and the feedforward approach for the landmarks part. The Hybrid Indirect approach maintains the benefits of the indirect formulation, i.e., it reduces the size of anchor frames stored in the vector state and it allows the representation of rotations with minimal representations avoiding singularities in the covariance matrix, but it limits the complexity of the delta reset step to $O(n)$, i.e., equivalent to a prediction step.

The use of the Hybrid Indirect EKF-SLAM has positive effects on the CI-SLAM framework too. First, if a standard EKF formulation is used, framed parameterizations result in singular covariance matrix due to the presence of quaternions. Although this issue can be solved by considering the covariance of the last three elements of quaternions during the *back propagation* step of the CI-SLAM, ax explained in Section 6.4.7.3, when non small rotations are involved, i.e., during a *loop closure* step, this approach fails. The indirect formulation of the EKF SLAM problem solves natively this issue by maintaining in the state vector only local rotations. The second reason lies in the need for maintaining the same nominal values between landmarks shared by different maps, to ensure the proper operation of the back propagation step of the CI-SLAM. The Hybrid Indirect EKF-SLAM approach copes with this requirement, since landmarks are treated with the feedforward approach, i.e., delta values are never integrated in the nominal values.

The particular structure of the EKF resulting from the use of framed parameterizations allows to extract information from the filter estimate, perform a refinement of the map through Bundle Adjustment techniques and reinsert the result of the optimization in the filter. Bundle Adjustment is performed on a graph that is constructed directly from the EKF estimate (i.e., the mean vector and the covariance matrix), as described in Section 8.1, before performing a map transition procedure of the CI-SLAM system. The particular structure of the graph eases the task of insert the results of the optimization process in the EKF solution (see Section 8.3). In particular, the optimization module has been developed to work on the Hybrid Indirect EKF-SLAM system with the FHP and FIS parameterizations. Particular care has been taken in the development of this module in order to obtain a plugin of the CI-SLAM framework: the module treats properly the monocular and the multi camera case, both when odometric data are available or not; moreover, special care has been taken in the construction of optimization graphs corresponding to maps successive to the first, as described in Section 8.2.7.

The implementation of the system described in this thesis has shown good performances, being able to operate in real time on monocular and stereo real dataset. The results on real datasets show that different parameterizations performs differently, although they show very similar performances in the NEES analysis on simulated data. This has been observed in [93] too, although experiments on real datasets have been conducted in a different setup, i.e., with a visual odometry system in a robocentric formulation. The addition of the local bundle adjustment on each submap improves the solution adding a computational cost that does not prevent the system from operating in real time on a standard PC.

As a conclusion, lets spend some words on the debate between the usage of filtering techniques for SLAM or to abandon them in favor of non linear optimization strategies. The recent trends in this research fields, strongly motivated by [96] and [98] tip the scale in favor of the non linear optimization techniques. We do not want to enter here in theoretical aspects regarding the comparison of the two solutions, but we want to underline some aspects that in our opinion motivates to push forward research in the filter based Visual SLAM. A first reason resides in the ability for the filter based visual SLAM system to drive the matching process of visual features in images, thanks to the *active search* mechanism based on measurement prediction in terms of mean and covariances, and to the ability of discarding outliers from the estimation process (e.g., with the 1Point RANSAC technique presented in [16] and used in this work). The feature matching and tracking process is usually discarded when comparing filter based and optimization based visual SLAM techniques (e.g., data association is assumed known in [98]), but it plays a key role in the estimation process, since the presence of outliers in the measurement process negatively affects the ability of trajectory and map estimation. The use of filtering techniques allows to manage the operation of tracking features and discarding outliers with cheap and easy solutions that run on standard CPU, in contrast with the request for complex algorithm of feature tracking based on parallel execution on graphical processing units (GPU) (e.g., in [95] the GPU based implementation of [79] is used). On the other hand, the filter driven matching process may results in monolithic structure that prevents from the development of a modular and well engineered software architecture based on available implementations (e.g., a feature tracker has to cope with elliptical research area that are not commonly considered in standard algorithms). A second reason that motivates the research in the filter based visual SLAM is the proven ability of operate in real time on standard computer architecture, that open the door to an effective deployment of visual SLAM applications on real robots equipped with low power computers or hand held devices based on micro controllers, although the introduction of low cost portable devices with very high performances may tip the scale further in favor of optimization based algorithms. As usual, when moving from pure research issues to engineered solutions some trade off needs to be found and we believe filtering techniques could rise from their current unpopular status and gather an important role in some practical application.

10.2 Future Works

We can spot numerous cues for further development of the proposed system. First of all, the problem can be extended to an auto-calibration setup, as originally proposed in [91] and [89], both for the CI-SLAM framework and for the CIBA SLAM framework. The only additional requirement regards the addition of parameters subject to calibration in the EKF state and in the optimization graph and their use in the measurement and update process of the EKF machinery and the definition of proper edges in the optimization graph.

To limit further the inconsistency due to the linearizations performed in the EKF system a robocentric approach can be applied, as done in [16]. This increases the complexity of the EKF-SLAM system, since each landmark has to be transformed to a new reference system at each iteration, resulting in an additional complexity of $O(n^2)$ that slow down the performances. Despite this, robocentric approach has been demonstrated able to run in real time on small maps and consequently it has been succesfully used in Visual Odometry applications. We can imagine to use the Conditional Independent submapping framework, properly adapted to the robocentric formulation of the SLAM problem, to perform a robocentric large scale SLAM. The optimization performed by the CIBA SLAM system can be easily adapted to this formulation too, and we can imagine to develop a new instance of the system presented in this work.

In order to realize a more complete SLAM system we have to introduce in our implementation a *loop detector* functionality that trigger the execution of a loop closure step of the CI-SLAM framework, as done in [77]. Moreover, the system has to be extended with the *map reuse* procedure provided by the CI-SLAM framework that have not be treated in this work. This will results in a complete visual SLAM system that may be deployed on a real robot to operate in real time.

To extend the multi camera system feature, some work on omnidirectional cameras has been done in our research group [81] but it has not been described in this thesis. The results of Monocular SLAM with omnidirectional camera are very promising and this can be explained with the richness of information provided by a $360°$ view of the environment. We expect that a multicamera system that involves both perspective and omnidirectional cameras may results in a high precision Visual SLAM system.

From a pure research point of view we believe that a deep comparison between filter based and optimization based visual SLAM systems is still missing. The available comparisons focus mainly on pure computational complexity in simulated setup where data association is known and the environment is controlled (e.g., noises are really Gaussian, odometry does not present slippages, etc.), while a complete comparison that takes into account the feature matching problem, which is differently addressed by the two systems, outliers rejection mechanisms, unexpected and unmodeled noises has never been performed to the best of our knowledge.

Bibliography

[1] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

[2] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. In *Proc. of IEEE Intern. Conf. on Intelligent Robots and Systems*, pages 3562–3568, 2006.

[3] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110:346–359, 2008.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[6] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An atlas framework for scalable mapping. In *IEEE International Conference on Robotics and Automation*, pages 1899–1906, 2003.

[7] D.C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[8] C. Cadena, D. Galvez-Lopez, J. D. Tardos, and J. Neira. Robust place recognition with stereo sequences. *IEEE Transactions on Robotics*, 28(4):871–885, August 2012.

[9] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371, 2009.

[10] Simone Ceriani, Daniele Marzorati, Matteo Matteucci, Davide Migliore, and Domenico G. Sorrenti. On feature parameterization for ekf-based monocular slam. In *In proceedings of 18th World Congress of the International Federation of Automatic Control (IFAC)*, pages 6829–6834, August 2011.

[11] Ward Cheney and David Kincaid. *Numerical Mathematics and Computing*. International Thomson Publishing, 4th edition, 1998.

[12] Kok Seng Chong and Lindsay Kleeman. Feature-based mapping in real, large scale environments using an ultrasonic array. *I. J. Robotic Res.*, 18(1):3–19, 1999.

[13] Gabe Sibley Christopher Mei and Paul Newman. Closing loops without places. In *International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010. 10.

[14] J. Civera, A.J. Davison, and J.M.M. Montiel. Inverse depth to depth conversion for monocular slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2778 – 2783, april 2007.

[15] Javier Civera, Andrew J. Davison, and José María Martínez Montiel. *Structure from Motion using the Extended Kalman Filter*, volume 75 of *Springer Tracts in Advanced Robotics*. Springer, 2012.

[16] Javier Civera, Oscar G. Grasa, Andrew J. Davison, and J. M. M. Montiel. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[17] Laura Clemente, Andrew Davison, Ian Reid, José Neira, and Juan Domingo Tardós. Mapping large loops with a single hand-held camera. In *Proc. Robotics: Science and Systems Conference*, June 2007.

[18] D.M Cole and P. M. Newman. Using laser range data for 3d SLAM in outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Orlando Florida USA, May 2006.

[19] A.J. Davison, Y. González Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon*, July 2004.

[20] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1403–, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Andrew J. Davison and David Murray. Mobile robot localisation using active vision. In Hans Burkhardt and Bernd Neumann, editors, *Computer Vision - ECCV'98*, volume 1407 of *Lecture Notes in Computer Science*, pages 809–825. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0054781.

[22] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[23] James Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, 2006.

[24] H. Durrant-Whyte, D. Rye, and E. Nebot. Localisation of automatic guided vehicles. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The 7th International Symposium (ISRR'95)*, pages 613–625. Springer Verlag, 1996.

[25] H. F. Durrant-Whyte. Uncertain Geometry in Robotics. *IEEE Trans. Robot. & Autom.*, 4(1):23–31, February 1988.

[26] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.

[27] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the rms titanic with slam information filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[28] John Folkesson and Henrik Christensen. Graphical SLAM - A Self-Correcting Map. In *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 383–390, 2004.

[29] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 1 edition, 2002.

[30] J. Funda and R.P. Paul. A comparison of transforms and quaternions in robotics. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 886–891 vol.2, apr 1988.

[31] Dorian Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[32] Dorian Galvez-Lopez and Juan D. Tardos. Real-time loop detection with bags of binary words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 51 –58, sept. 2011.

[33] Philip E. Gill and Walter Murray. Algorithms for the Solution of the Nonlinear Least-Squares Problem. *SIAM Journal on Numerical Analysis*, 15(5):977–992, 1978.

[34] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering : Theory and Practice Using MATLAB*. Wiley-Interscience, 2 edition, January 2001.

[35] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intell. Transport. Syst. Mag.*, pages 31–43, 2010.

[36] Jose Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001.

[37] W.R. Hamilton. *Lectures on Quaternions*. Hodges and Smith, 1853.

[38] W.R. Hamilton. *Elements of Quaternions*. Longmans, Green, & Company, 1866.

[39] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[40] Eugene Hecht. *Optics (4th Edition)*. Addison Wesley, 4 edition, 2001.

[41] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 1106–, Washington, DC, USA, 1997. IEEE Computer Society.

[42] KinLeong Ho and Paul Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74:261–286, 2007.

[43] E. Imre, M.-O. Berger, and N. Noury. Improved inverse-depth parameterization for monocular simultaneous localization and mapping. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 381 –386, may 2009.

[44] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[45] Jong-Hyuk Kim and S. Sukkarieh. Airborne simultaneous localisation and map building. volume 1, pages 406–411 vol.1, 2003.

[46] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[47] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, October 2009.

[48] K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *Robotics, IEEE Transactions on*, 24(5):1066 –1077, oct. 2008.

[49] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[50] J. Kurlbaum and Udo Frese. A benchmark dataset for data association. Technical report, SFB/TR 8, 2009.

[51] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3+, 1991.

[52] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Boston: Kluwer Academic Publishers, 1992.

[53] Hui-Ping LI, De-Min XU, Fu-Bin ZHANG, and Yao YAO. Consistency analysis of ekf-based slam by measurement noise and observation times. *Acta Automatica Sinica*, 35(9):1177 – 1184, 2009.

[54] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[55] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[56] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[57] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for Non-Linear Least Squares Problems (2nd ed.), 2004.

[58] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[59] I. Mahon, S.B. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based slam using visual loop closures. *Robotics, IEEE Transactions on*, 24(5):1002 –1014, oct. 2008.

[60] Andras Majdik, Dorian Galvez-Lopez, Gheorghe Lazea, and Jose A. Castellanos. Adaptive appearance based loop-closing in heterogeneous environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1256 –1263, sept. 2011.

[61] Ruben Martinez-Cantin and José A. Castellanos. Bounding uncertainty in ekf-slam: the robocentric local approach. In *ICRA*, pages 430–435. IEEE, 2006.

[62] D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti. Monocular slam with inverse scaling parametrization. In R. Frile M. Everingham, C.J. Needham, editor, *Proc.s of 2008 British Machine Vision Conf. (BMVC 2008)*, pages 945–954, 2008.

[63] D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti. On the use of inverse scaling in monocular slam. In *Proc. of IEEE Intern. Conf. on Robotics and Automation*, pages 2030–2036, 2009.

[64] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615 –1630, oct. 2005.

[65] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[66] Oscar Martinez Mozos, Arturo Gil, Monica Ballesta, and Oscar Reinoso. Interest point detectors for visual SLAM. In Daniel Borrajo, Luis Castillo, and Juan Manuel Corchado, editors, *Current Topics in Artificial Intelligence*, volume 4788 of *Lecture Notes in Computer Science*, pages 170–179. Springer Berlin / Heidelberg, Germany, 2007.

[67] R. Munguia and A. Grau. Monocular slam for visual odometry. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 1 –6, oct. 2007.

[68] J. Neira and J.D. Tardós. Data Association in Stochastic Mapping Using the Joint Compatibility Test. *IEEE Transactions on Robotics and Automation*, Vol. 17(No. 6):pp. 890 – 897, December 2001.

[69] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180 –1187, may 2006.

[70] P. Newman and Kin Ho. Slam-loop closing with visually salient features. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 635 – 642, april 2005.

[71] Paul Newman and John J. Leonard. Pure range-only subsea slam. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Taiwan, September 2003.

[72] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. Mc-Graw Hill, 1984.

[73] M.P. Parsley and S.J. Julier. Avoiding negative depth in inverse depth bearing-only slam. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2066 –2071, sept. 2008.

[74] T. Pietzsch. Efficient feature parameterisation for visual slam using inverse depth bundles. In *Proceedings of the British Machine Vision Conference*, pages 5.1–5.10. BMVA Press, 2008. doi:10.5244/C.22.5.

[75] P. Pinies, T. Lupton, S. Sukkarieh, and J.D. Tardos. Inertial aiding of inverse depth slam using a monocular camera. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2797 –2802, april 2007.

[76] Pedro Piniés. *SLAM in Large Environments with Wearable Sensors*. PhD thesis, Universidad de Zaragoza (Spain), March 2009.

[77] Pedro Piniés, Lina María Paz, Dorian Gálvez-López, and Juan D. Tardós. Ci-graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system. *Journal of Field Robotics*, 27(5):561–586, 2010.

[78] Pedro Pinies and Juan Doming Tardos. Large-scale slam building conditionally independent local maps: Application to monocular vision. *Robotics, IEEE Transactions on*, 24(5):1094 –1106, oct. 2008.

[79] T. Pock, M. Unger, D. Cremers, and H. Bischof. Fast and exact solution of Total Variation models on the GPU. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –8, june 2008.

[80] W. D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *International Conference on Intelligent RObots and Systems - IROS*, 1993.

[81] Bacciocchi Roberto. SLAM con camera omnidirezionale all'interno del framework MoonSlam. Master's thesis, Politecnico di Milano (Italy), 2012.

[82] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[83] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[84] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *Int. J. Comput. Vision*, 37(2):151–172, June 2000.

[85] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[86] M. D. Shuster. Survey of attitude representations. *Journal of the Astronautical Sciences*, 41:439–517, October 1993.

[87] R. Smith and P Cheeseman. On the representation and estimation of spatial uncertainty. *Int Journal of Robotics Research*, 5:56–68, 1987.

[88] R. Smith, M. Self, and P. Cheeseman. Autonomous robot vehicles. chapter Estimating uncertain spatial relationships in robotics, pages 167–193. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[89] Joan Solà. *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.* PhD thesis, Institut National Polytechnique de Toulouse, 2007.

[90] Joan Solà. Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3513 –3518, may 2010.

[91] Joan Solà, André Monin, and Michel Devy. BiCamSLAM: Two times mono is more than stereo. In *ICRA*, pages 4795–4800. IEEE, 2007.

[92] Joan Solà, André Monin, Michel Devy, and Thomas Lemaire. Undelayed initialization in bearing only slam. In *IROS*, pages 2499–2504. IEEE, 2005.

[93] Joan Sola, Teresa Vidal-Calleja, Javier Civera, and J. M. M. Montiel. Impact of landmark parametrization on monocular ekf-slam with points and lines. *International Journal of Computer Vision*, accepted for publication:–, 2011.

[94] Michael Spivak. *Calculus on manifolds. A modern approach to classical theorems of advanced calculus.* W. A. Benjamin, Inc., New York-Amsterdam, 1965.

[95] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[96] H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657 –2664, may 2010.

[97] Hauke Strasdat, Andrew J. Davison, J. M. M. Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 2352–2359. IEEE, 2011.

[98] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Editors choice article: Visual slam: Why filter? *Image Vision Comput.*, 30(2):65–77, February 2012.

[99] John Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM Review*, 6(4):pp. 422–430, 1964.

[100] N. Sunderhauf, S. Lange, and P. Protzel. Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1 –6, sept. 2007.

[101] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[102] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[103] Sebastian Thrun and Michael Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *Internation Journal On Robotic Research*, 25(5):403–430, 2006.

[104] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK, 2000. Springer-Verlag.

[105] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, 2008.

[106] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.

[107] Liang Zhao, Shoudong Huang, Lei Yan, and G. Dissanayake. Parallax angle parametrization for monocular slam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3117 –3124, may 2011.

Part III

Appendices

APPENDIX A

Consistency Analysis

In this appendix we report the complete results of the consistency analysis performed in simulated experiments of Chapter 9. Refer to Section 9.1.1.1 and Section 9.1.1.2 for details on the experiments, respectively for the monocular and the stereo case. Table A.1 reports the results of the average inconsistency for the stereo simulated experiments when the viewing ray of the landmarks is considered exact. Table A.2 and Table A.3 report respectively the consistency analysis and the average inconsistency value of monocular experiments with noised initialization of the initial viewing ray; Table A.4 and Table A.5 refer to monocular experiments with noised initialization of the initial viewing ray using a high resolution camera; Table A.6 and Table A.7 refer to stereo experiments with noised initialization of the initial viewing ray; Table A.8 and Table A.9 refer to stereo experiments with noised initialization of the initial viewing ray using a high resolution camera.

Table A.1: *Stereo - Mean inconsistency, exact initialization*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	342.9	**4.9**	5.4	5.2	5.2	4.1	**3.8**	0.3	0.3
8.b	6.1	**0.3**	**0.3**	**0.3**	**0.3**	0.4	0.5	–	–
8.c	20.3	**0.3**	**0.3**	**0.3**	**0.3**	0.4	0.4	–	–
9.a	251.6	3.5	2.9	**2.8**	**2.8**	**1.9**	2.2	–	–
9.b	4.4	0.4	**0.3**	**0.3**	**0.3**	0.6	0.6	–	–
9.c	17.3	0.4	**0.3**	**0.3**	**0.3**	0.5	0.5	–	–
10.a	371.0	6.2	6.2	6.0	**5.9**	**1.5**	**1.5**	2.3	2.2
10.b	6.0	0.3	0.3	**0.2**	**0.2**	0.7	0.6	–	–
10.c	25.7	**0.3**	**0.3**	0.4	**0.3**	0.5	0.5	–	–
11.a	551.5	**5.9**	6.5	6.7	7.1	2.2	**2.1**	5.0	5.0
11.b	8.1	0.3	**0.2**	**0.2**	0.3	0.4	0.5	–	–
11.c	33.8	**0.2**	0.3	0.3	0.4	0.5	0.5	–	–
12.a	154.4	3.9	2.7	**2.6**	**2.6**	0.7	**0.6**	0.1	0.1
12.b	1.0	0.3	0.4	0.3	**0.2**	0.5	0.5	–	–
12.c	9.5	0.4	**0.3**	**0.3**	0.5	0.5	0.5	–	–
13.a	180.9	**5.9**	6.2	6.4	6.5	0.8	**0.7**	0.8	0.8
13.b	1.6	0.5	**0.4**	0.5	**0.4**	0.8	0.7	–	–
13.c	12.3	0.6	**0.5**	0.6	0.7	0.7	0.7	–	–
14.a	150.0	9.0	**7.9**	8.3	**7.9**	**0.8**	0.9	0.9	0.9
14.b	2.1	0.6	**0.4**	0.5	0.7	0.9	0.9	–	–
14.c	13.4	**0.8**	1.1	1.1	1.5	**0.6**	1.0	–	–

Table A.2: *Monocular - Consistency analysis, noised initialization*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	Cons %	1	3	4	**5**	4	0	0	0	0
	Opt %	99	97	**96**	**96**	**96**	100	100	100	100
1.b	Cons %	3	37	36	38	**40**	0	0	1	1
	Opt %	97	63	64	62	**60**	100	100	99	99
1.c	Cons %	3	20	**21**	**21**	**21**	0	0	1	1
	Opt %	97	81	**79**	80	80	100	100	99	99
2.a	Cons %	3	25	29	**30**	**30**	0	0	0	0
	Opt %	98	75	71	**70**	**70**	100	100	100	100
2.b	Cons %	8	65	76	**78**	76	0	0	1	1
	Opt %	92	35	24	**22**	24	100	100	99	99
2.c	Cons %	5	69	76	**77**	**77**	0	0	1	1
	Opt %	95	31	24	**23**	**23**	100	100	99	99
3.a	Cons %	4	15	**17**	**17**	**17**	0	0	0	0
	Opt %	96	85	**83**	**83**	84	100	100	100	100
3.b	Cons %	5	41	**62**	**62**	59	0	0	0	0
	Opt %	96	59	**38**	**38**	42	100	100	100	100
3.c	Cons %	4	21	**48**	36	44	0	0	0	0
	Opt %	97	80	**52**	64	56	100	100	100	100
4.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
4.b	Cons %	**1**	**1**	**1**	**1**	**1**	0	0	0	0
	Opt %	100	**99**	**99**	**99**	**99**	100	100	100	100
4.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
5.a	Cons %	0	34	37	46	**52**	0	0	0	0
	Opt %	100	66	63	54	**48**	100	100	100	100
5.b	Cons %	1	33	49	40	**56**	0	0	0	0
	Opt %	99	67	51	60	**44**	100	100	100	100
5.c	Cons %	1	**61**	33	40	52	0	0	0	0
	Opt %	99	**39**	67	60	48	100	100	100	100
6.a	Cons %	0	12	**14**	10	10	0	0	0	0
	Opt %	100	88	**86**	90	90	100	100	100	100
6.b	Cons %	0	**29**	25	21	26	0	0	0	0
	Opt %	100	**71**	75	79	74	100	100	100	100
6.c	Cons %	0	16	10	15	**17**	0	0	0	0
	Opt %	100	84	90	85	**83**	100	100	100	100
7.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
7.b	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
7.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100

Table A.3: *Monocular - Mean inconsistency, noised initialization*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	203.7	2.6	1.9	1.9	**1.8**	107.9	107.3	38.3	38.2
1.b	45.1	0.6	0.6	0.6	**0.5**	102.4	99.5	36.3	35.8
1.c	68.4	1.0	**0.8**	**0.8**	**0.8**	93.9	101.7	36.9	37.7
2.a	124.9	2.4	2.2	**2.1**	**2.1**	76.9	76.7	33.8	33.5
2.b	19.3	0.4	0.4	**0.3**	**0.3**	73.1	74.5	31.9	31.7
2.c	32.4	**0.4**	**0.4**	**0.4**	**0.4**	72.3	72.7	30.7	30.6
3.a	162.1	1.2	**1.1**	**1.1**	1.2	153.6	156.7	56.2	56.3
3.b	42.5	0.5	**0.4**	**0.4**	**0.4**	170.0	174.1	62.4	62.8
3.c	76.4	0.9	**0.5**	0.6	**0.5**	141.7	144.6	59.3	59.2
4.a	1018.5	12.9	**12.4**	12.9	12.7	215.7	212.3	64.3	64.0
4.b	304.7	12.1	**11.9**	12.1	12.1	217.2	216.9	63.9	64.2
4.c	393.2	13.0	**12.6**	12.7	**12.6**	213.6	212.3	58.0	58.1
5.a	78.5	0.8	0.7	**0.6**	**0.6**	120.2	127.2	79.3	78.8
5.b	14.2	0.8	**0.6**	0.7	**0.6**	146.9	138.0	84.7	87.4
5.c	33.1	**0.5**	0.7	0.6	**0.5**	131.2	131.4	97.8	92.5
6.a	185.8	2.4	**1.9**	2.6	2.4	114.3	107.0	45.3	44.6
6.b	41.4	1.7	1.6	1.9	**1.2**	124.5	126.8	63.0	62.2
6.c	76.5	1.7	2.1	1.8	**1.5**	133.7	131.2	88.7	85.5
7.a	710.0	23.9	**22.6**	23.8	24.1	161.4	166.8	43.2	41.6
7.b	303.2	22.2	21.9	**21.7**	23.6	204.0	207.3	63.5	57.4
7.c	595.4	23.4	23.3	23.1	**22.8**	186.5	210.1	71.6	67.5

Table A.4: *Monocular - Consistency analysis, noised initialization, high res. camera*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
1.b	Cons %	1	**4**	**4**	**4**	**4**	0	0	0	0
	Opt %	99	**96**	**96**	**96**	**96**	100	100	100	100
1.c	Cons %	0	**2**	**2**	**2**	**2**	0	0	0	0
	Opt %	100	**98**	**98**	**98**	**98**	100	100	100	100
2.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
2.b	Cons %	3	66	**73**	72	**73**	0	0	0	0
	Opt %	97	34	**27**	28	**27**	100	100	100	100
2.c	Cons %	2	61	**63**	**63**	62	0	0	0	0
	Opt %	98	39	**37**	**37**	38	100	100	100	100
3.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
3.b	Cons %	0	**3**	**3**	**3**	**3**	0	0	0	0
	Opt %	100	**98**	**98**	**98**	**98**	100	100	100	100
3.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
4.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
4.b	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
4.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
5.a	Cons %	0	**2**	1	**2**	**2**	0	0	0	0
	Opt %	100	**98**	99	**98**	**98**	100	100	100	100
5.b	Cons %	0	32	29	**33**	30	0	0	0	0
	Opt %	100	68	71	**67**	70	100	100	100	100
5.c	Cons %	0	**18**	**18**	15	**18**	0	0	0	0
	Opt %	100	**82**	**82**	85	**82**	100	100	100	100
6.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
6.b	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
6.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
7.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
7.b	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
7.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100

Table A.5: *Monocular - Mean inconsistency, noised initialization, high resolution camera*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
1.a	7726.6	**11.7**	12.2	12.2	12.2	130.9	134.3	35.5	35.5
1.b	215.3	2.7	2.7	2.7	**2.6**	127.0	125.6	32.4	32.6
1.c	524.8	**2.6**	**2.6**	**2.6**	**2.6**	131.4	130.7	32.1	32.1
2.a	1632.1	5.2	**4.6**	**4.6**	**4.6**	83.1	83.0	29.7	29.7
2.b	48.3	0.6	**0.5**	**0.5**	**0.5**	104.1	104.8	28.7	28.7
2.c	103.6	0.7	**0.6**	**0.6**	**0.6**	116.3	116.6	29.7	29.6
3.a	3879.8	12.3	**12.0**	**12.0**	12.1	223.3	222.3	54.0	54.0
3.b	449.6	5.9	**5.8**	**5.8**	5.9	218.1	220.7	55.9	55.8
3.c	899.7	**7.9**	8.1	8.1	8.0	218.5	218.0	55.3	55.2
4.a	16515.3	59.3	59.0	59.0	**56.9**	645.9	644.9	87.1	87.1
4.b	7219.0	62.2	62.1	62.1	**61.9**	777.6	778.0	96.7	96.6
4.c	7953.2	59.9	**59.7**	59.8	59.9	771.4	772.6	94.5	94.4
5.a	907.8	2.7	3.4	3.2	**2.5**	119.4	120.4	41.3	40.2
5.b	56.7	1.3	1.5	**1.2**	1.4	145.5	143.0	48.4	48.6
5.c	129.3	1.4	1.6	1.7	**1.3**	181.8	180.1	55.2	57.8
6.a	3683.6	**16.2**	16.6	16.6	16.9	138.5	136.2	47.8	48.1
6.b	486.1	15.1	15.3	15.2	**14.7**	168.1	163.7	46.9	44.3
6.c	974.4	17.6	**15.8**	16.1	15.9	196.9	196.1	43.6	43.6
7.a	24840.5	110.0	104.7	**104.6**	105.0	449.4	441.8	97.1	96.1
7.b	4899.7	119.0	118.6	120.2	**117.6**	574.3	570.6	110.3	111.5
7.c	9548.8	**115.2**	116.7	116.5	116.7	583.2	582.9	120.7	120.2

Table A.6: *Stereo - Consistency analysis, noised initialization*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	Cons %	0	**2**	1	1	1	0	0	0	0
	Opt %	100	**98**	99	99	99	100	100	100	100
8.b	Cons %	13	77	86	83	**87**	0	0	0	0
	Opt %	87	23	14	17	**13**	100	100	100	100
8.c	Cons %	4	68	78	**79**	77	0	0	0	0
	Opt %	96	32	23	**21**	24	100	100	100	100
9.a	Cons %	1	10	**11**	**11**	10	0	0	0	0
	Opt %	99	90	**89**	**89**	90	100	100	100	100
9.b	Cons %	13	85	**91**	**91**	**91**	0	0	1	1
	Opt %	88	15	10	10	**9**	100	100	100	100
9.c	Cons %	5	82	88	88	**89**	0	0	1	1
	Opt %	95	18	12	12	**11**	100	100	100	100
10.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
10.b	Cons %	5	91	85	85	**92**	0	0	1	1
	Opt %	95	9	16	15	**8**	100	100	100	100
10.c	Cons %	3	**89**	83	87	85	0	0	0	0
	Opt %	97	**11**	17	13	15	100	100	100	100
11.a	Cons %	0	**1**	0	0	0	0	0	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
11.b	Cons %	3	88	**92**	**92**	90	0	0	0	0
	Opt %	97	12	**7**	8	10	100	100	100	100
11.c	Cons %	1	78	**81**	80	78	0	0	0	0
	Opt %	99	22	**18**	19	22	100	100	100	100
12.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
12.b	Cons %	8	85	72	85	**88**	0	0	0	0
	Opt %	92	14	28	15	**12**	100	100	100	100
12.c	Cons %	3	78	82	**84**	73	0	0	0	0
	Opt %	97	21	18	**16**	27	100	100	100	100
13.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
13.b	Cons %	3	26	57	51	**58**	0	0	0	0
	Opt %	97	74	43	49	**42**	100	100	100	100
13.c	Cons %	1	**31**	27	26	28	0	0	0	0
	Opt %	99	**69**	73	74	72	100	100	100	100
14.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
14.b	Cons %	3	21	**25**	23	23	0	0	0	0
	Opt %	97	79	**75**	77	77	100	100	100	100
14.c	Cons %	1	**23**	16	16	12	0	0	0	0
	Opt %	99	**77**	84	84	88	100	100	100	100

Table A.7: *Stereo - Mean inconsistency, noised initialization*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	286.2	**5.3**	5.4	**5.3**	5.4	187.1	188.4	72.2	72.3
8.b	7.2	0.5	0.5	0.5	**0.4**	213.6	210.8	95.2	96.8
8.c	19.6	**0.6**	0.7	**0.6**	**0.6**	195.4	196.5	93.4	94.1
9.a	251.3	3.5	**3.3**	**3.3**	3.4	167.3	176.1	67.4	67.4
9.b	6.4	0.5	**0.4**	**0.4**	**0.4**	226.9	224.7	73.5	74.1
9.c	23.9	**0.4**	0.5	0.5	0.5	223.3	225.0	76.8	77.1
10.a	366.0	4.9	**4.8**	5.0	**4.8**	91.2	88.9	50.5	50.6
10.b	9.1	**0.3**	0.4	0.4	**0.3**	160.6	161.4	57.4	57.0
10.c	31.9	0.5	0.5	0.5	**0.4**	160.4	165.6	57.2	54.6
11.a	510.2	6.5	**6.1**	**6.1**	6.3	137.4	126.1	44.2	45.4
11.b	9.5	**0.3**	**0.3**	**0.3**	0.4	179.8	183.4	52.5	52.5
11.c	32.6	0.5	**0.4**	0.5	0.5	200.8	196.0	56.1	56.5
12.a	157.7	4.1	**3.8**	4.3	4.4	223.4	241.0	137.7	131.4
12.b	1.9	**0.3**	0.5	0.4	0.4	300.0	336.3	192.9	184.1
12.c	11.3	0.5	**0.4**	**0.4**	0.5	370.2	304.5	183.3	177.2
13.a	145.4	6.2	**5.9**	6.1	6.0	231.1	230.6	107.8	115.6
13.b	3.0	1.0	**0.6**	**0.6**	**0.6**	289.1	330.9	157.8	132.1
13.c	14.8	**0.9**	1.1	1.0	1.1	247.3	266.0	139.9	139.0
14.a	178.1	10.0	8.2	**7.5**	7.7	209.9	206.0	80.7	81.5
14.b	4.0	1.3	1.3	**1.2**	1.4	244.3	274.4	86.1	81.5
14.c	13.5	**1.1**	1.2	1.5	1.5	219.3	231.6	82.5	78.2

Table A.8: *Stereo - Consistency analysis, noised initialization, high resolution camera*

#		IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
8.b	Cons %	0	19	18	18	**20**	0	0	0	0
	Opt %	100	81	82	82	**80**	100	100	100	100
8.c	Cons %	0	**1**	**1**	**1**	**1**	0	0	0	0
	Opt %	100	**99**	**99**	**99**	**99**	100	100	100	100
9.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
9.b	Cons %	2	20	**25**	**25**	23	0	0	0	0
	Opt %	98	80	**75**	**75**	77	100	100	100	100
9.c	Cons %	0	**1**	**1**	**1**	**1**	0	0	0	0
	Opt %	100	**99**	**99**	**99**	**99**	100	100	100	100
10.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
10.b	Cons %	0	**1**	**1**	**1**	**1**	0	0	0	0
	Opt %	100	**99**	**99**	**99**	**99**	100	100	100	100
10.c	Cons %	0	**11**	**11**	**11**	**11**	0	0	0	0
	Opt %	100	**89**	**89**	**89**	**89**	100	100	100	100
11.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
11.b	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
11.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
12.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
12.b	Cons %	0	17	16	15	**18**	0	0	0	0
	Opt %	100	83	84	85	**82**	100	100	100	100
12.c	Cons %	0	0	**2**	**2**	1	0	0	0	0
	Opt %	100	100	**98**	**98**	99	100	100	100	100
13.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
13.b	Cons %	0	7	**8**	7	**8**	0	0	0	0
	Opt %	100	93	**92**	93	**92**	100	100	100	100
13.c	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
14.a	Cons %	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0
	Opt %	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
14.b	Cons %	1	9	**10**	**10**	9	0	0	0	0
	Opt %	99	91	**90**	**90**	91	100	100	100	100
14.c	Cons %	0	**4**	**4**	**4**	**4**	0	0	0	0
	Opt %	100	**96**	**96**	**96**	**96**	100	100	100	100

Table A.9: *Stereo - Mean inconsistency, noised initialization, high resolution camera*

#	IS	UID	AHP	FHP	FHP$_i$	FIS0	FIS0$_i$	FIS	FIS$_i$
8.a	1414.4	**39.6**	40.5	40.7	42.6	193.2	198.9	59.0	59.9
8.b	9.1	3.1	**3.0**	**3.0**	3.2	214.6	220.3	61.9	62.7
8.c	29.3	**7.1**	**7.1**	**7.1**	**7.1**	225.3	204.7	62.0	62.1
9.a	1067.2	22.9	20.6	20.6	**20.2**	169.0	186.8	70.6	72.1
9.b	9.8	**1.9**	**1.9**	**1.9**	**1.9**	209.0	200.1	77.1	77.5
9.c	38.4	**4.2**	4.3	4.3	4.3	184.6	178.7	69.1	66.6
10.a	5511.7	115.5	98.5	**98.3**	101.2	150.9	154.7	52.7	53.7
10.b	32.2	13.3	**13.1**	**13.1**	**13.1**	181.7	181.6	48.4	48.4
10.c	34.5	3.9	3.9	3.9	**3.8**	181.6	180.7	57.3	55.4
11.a	3487.3	72.4	**68.4**	68.5	69.8	191.4	197.3	29.1	29.2
11.b	36.9	14.4	14.4	14.4	**14.3**	217.3	215.5	35.5	35.5
11.c	113.4	27.1	26.8	26.8	**26.6**	201.5	202.3	36.4	36.4
12.a	1292.5	13.4	**12.9**	**12.9**	13.7	256.8	267.3	127.3	137.7
12.b	5.0	**1.7**	1.9	1.9	2.1	252.1	240.5	93.7	94.9
12.c	31.6	**5.7**	5.8	6.0	5.8	263.1	266.4	89.1	93.8
13.a	2827.6	39.0	**38.0**	**38.0**	39.7	361.8	376.5	65.8	65.0
13.b	4.9	2.5	**2.4**	**2.4**	2.8	260.9	247.8	64.2	64.6
13.c	26.9	6.6	6.6	**6.5**	7.3	194.2	209.6	62.4	61.2
14.a	2718.5	72.9	71.4	**71.1**	74.5	259.2	236.9	37.9	41.8
14.b	4.6	**2.8**	**2.8**	**2.8**	3.5	229.2	217.7	37.7	37.8
14.c	21.1	6.6	**6.5**	**6.5**	7.3	223.6	236.1	39.4	42.6