



SAPIENZA  
UNIVERSITÀ DI ROMA

# High-Level Environment Representations for Mobile Robots

Sapienza University of Rome

Dottorato di Ricerca in Engineering in Computer Science – XXX Ciclo

Candidate

Federico Nardi

ID number 1463717

Thesis Advisors

Prof. Giorgio Grisetti

Prof. Daniele Nardi

February 2019

Thesis defended on 22 February 2019  
in front of a Board of Examiners composed by:  
Prof. Riccardo Torlone (chairman)  
Prof. Alessandro Farinelli  
Prof. Paolo Prinetto

---

**High-Level Environment Representations for Mobile Robots**

Ph.D. thesis. Sapienza – University of Rome

ISBN: 000000000-0

© 2019 Federico Nardi. All rights reserved

This thesis has been typeset by L<sup>A</sup>T<sub>E</sub>X and the Sapthesis class.

Version: January 21, 2019

Author's email: fnardi@diag.uniroma1.it

*Dedicated to  
who has made its writing possible and whoever will read it*



## Abstract

In most robotic applications we are faced with the problem of building a digital representation of the environment that allows the robot to autonomously complete its tasks. This internal representation can be used by the robot to plan a motion trajectory for its mobile base and/or end-effector. For most man-made environments we do not have a digital representation or it is inaccurate. Thus, the robot must have the capability of building it autonomously. This is done by integrating into an internal data structure incoming sensor measurements. For this purpose, a common solution consists in solving the Simultaneous Localization and Mapping (SLAM) problem. The map obtained by solving a SLAM problem is called “metric” and it describes the geometric structure of the environment. A metric map is typically made up of low-level primitives (like points or voxels). This means that even though it represents the shape of the objects in the robot workspace it lacks the information of which object a surface belongs to. Having an object-level representation of the environment has the advantage of augmenting the set of possible tasks that a robot may accomplish. To this end, in this thesis we focus on two aspects. We propose a formalism to represent in a uniform manner 3D scenes consisting of different geometric primitives, including points, lines and planes. Consequently, we derive a local registration and a global optimization algorithm that can exploit this representation for robust estimation. Furthermore, we present a Semantic Mapping system capable of building an *object-based* map that can be used for complex task planning and execution. Our system exploits effective reconstruction and recognition techniques that require no a-priori information about the environment and can be used under general conditions.



## Acknowledgments

*This is the section of the thesis devoted to gratitude. Gratitude is a wonderful yet complex feeling and, as all complex things in life, it takes practice to handle it. In this moment, all my gratitude goes to the members of the LabRoCoCo that I met during this adventure. Despite I joined late this group, I have been warmly welcomed. It took me a while to realize how lucky I was to end up in such a nice community, so I'm taking this chance to say it now. I could thank each of them separately but it would not be accurate, because each of them knows better than me what was his/her contribution to this work and how much patience they had to exercise while I made them upset with my bad attitude ("Sorry for that! I'm working on it!"). What I really want to do is to thank every single member of the lab for the lessons I learned from them. Apart from the technical aspects, I can say that I learnt from this group how to become a better person and cultivate the values of responsibility, honesty and respect.*





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Research Context . . . . .	1
1.2	Addressed issues . . . . .	2
1.3	Contributions . . . . .	4
<b>I</b>	<b>Basics</b>	<b>7</b>
<b>2</b>	<b>Related Work</b>	<b>9</b>
2.1	Geometric Registration . . . . .	9
2.1.1	Global Registration . . . . .	9
2.1.2	Local Registration . . . . .	10
2.1.3	Geometric Primitives Detection . . . . .	12
2.2	Simultaneous Localization and Mapping . . . . .	14
2.2.1	Graph-based SLAM . . . . .	14
2.2.2	SLAM with Geometric Primitives . . . . .	15
2.3	Semantic Mapping . . . . .	16
2.3.1	Semantic Information Extraction . . . . .	16
2.3.2	Map Construction . . . . .	17
2.3.3	Active Vision . . . . .	19
<b>3</b>	<b>Fundamentals</b>	<b>21</b>
3.1	State Estimation . . . . .	21
3.1.1	Bayesian Framework . . . . .	21
3.1.2	Gauss-Newton . . . . .	22
3.1.3	Smooth Manifolds . . . . .	24
3.2	Local Registration . . . . .	28
3.2.1	Iterative Closest Point . . . . .	28
3.2.2	Least-Squares Solution . . . . .	29
3.3	Global Optimization . . . . .	31
3.3.1	Multi-Point Registration . . . . .	31
3.3.2	Factor Graphs and Sparse Least-Squares . . . . .	32

<b>II</b>	<b>Metric maps</b>	<b>35</b>
<b>4</b>	<b>Taxonomy of Metric Map Representations</b>	<b>37</b>
4.1	Sparse Representation . . . . .	38
4.2	Dense Representation . . . . .	39
4.3	Volumetric Representation . . . . .	40
4.4	Object-based Representation . . . . .	41
4.5	Comparison . . . . .	41
<b>5</b>	<b>Unifying Local Registration Algorithms</b>	<b>43</b>
5.1	Generalized Local Registration . . . . .	43
5.1.1	Representation . . . . .	45
5.1.2	Transformation . . . . .	45
5.1.3	Distance . . . . .	46
5.1.4	Registration . . . . .	47
5.2	Front-End . . . . .	50
5.2.1	Detecting Matchables from RGB-D data . . . . .	50
5.2.2	Data Association . . . . .	51
5.3	Experimental Evaluation . . . . .	52
5.3.1	Synthetic Data . . . . .	52
5.3.2	Simulated and Real Data . . . . .	53
<b>6</b>	<b>Unifying Global Optimization Algorithms</b>	<b>57</b>
6.1	Multi-Primitive Registration . . . . .	58
6.1.1	State . . . . .	58
6.1.2	Error . . . . .	59
6.2	Front-End . . . . .	61
6.2.1	Pose Tracker . . . . .	62
6.2.2	Loop Detector . . . . .	62
6.3	Experimental Evaluation . . . . .	63
6.3.1	Synthetic Experiments . . . . .	63
6.3.2	Real-world Experiments . . . . .	64
<b>III</b>	<b>Semantic maps</b>	<b>67</b>
<b>7</b>	<b>Taxonomy of a Semantic Mapping System</b>	<b>69</b>
7.1	Perception . . . . .	69
7.2	Map Construction . . . . .	71
7.3	Action . . . . .	71
<b>8</b>	<b>Generating a Semantic Map</b>	<b>73</b>
8.1	System Overview . . . . .	74
8.1.1	Perception . . . . .	74
8.1.2	Map Construction . . . . .	76
8.1.3	Action . . . . .	80
8.2	Experimental Evaluation . . . . .	82
8.2.1	Constructing the Map . . . . .	84

---

8.2.2	Exploring the Environment . . . . .	85
<b>IV</b>	<b>Conclusions</b>	<b>89</b>
<b>9</b>	<b>Final Discussion</b>	<b>91</b>
9.1	Metric Maps . . . . .	91
9.2	Semantic Maps . . . . .	92
9.3	Future Directions . . . . .	92



# List of Figures

1.1	Example of environment reconstructions obtained with: (a) Velodyne (image from [175]) and (b) Kinect (image from [171]). . . . .	2
1.2	Convolutional Neural Network [81]: (a) AlexNet architecture and (b) recognition results. . . . .	3
4.1	Incremental map building process. . . . .	38
4.2	Example of sparse representations: (a) image from [22] (b) image from [76]. . . . .	39
4.3	Example of dense representations: (a) image from [102] (b) image from [170]. . . . .	39
4.4	Example of volumetric representations: (a) image from [103] (b) image from [64]. . . . .	40
4.5	Example of object-based representation, image from [132]. . . . .	41
5.1	A typical scenario addressable with the proposed representation: we want to register a moving scene (red) onto a fixed scene (black). The scenes are composed by points ( <b>pt</b> ), lines ( <b>l</b> ) and planes ( <b>pl</b> ). Green lines indicate the constraints between two geometric entities. The proposed representation allows to model both homogeneous ( <i>e.g.</i> line-line) and heterogeneous ( <i>e.g.</i> point-plane) constraints. . . . .	44
5.2	Front-end. When a new pair of images is available, we extract a set of matchables from raw data using the strategy outlined in Sec. 5.2.1, here we show only planes for more clarity. Subsequently, we find corresponding matchables (here, linked by a red line) between the newly generated scene and the “fixed” scene with the methodology in Sec. 5.2.2. Finally, a minimization is conducted to find the transform that best aligns the two scenes according to Sec. 5.1.4. . . . .	50
5.3	Planes extraction. From left to right and top to bottom: (a) input depth image, (b) estimated surface normals, (c) connected regions (in black), (d) detected planes (in red). . . . .	51
5.4	Synthetic Experiments - Error Evolution. In <i>solid blue</i> is shown the evolution of the error without noise, while in <i>dashed red</i> is reported the low-noise case and in <i>point-dashed yellow</i> the high-noise case. For each constraint, the left plot reports the Iterative Solver evolution, while the right plot shows the error after one Direct Solver iteration. . . . .	53
6.1	Applying a perturbation to the matchable direction. . . . .	59

6.2	Front-end for global optimization. . . . .	61
6.3	Visual comparison of the two factor graphs rendered by the <code>g2o_viewer</code> : (a) before optimization (b) after optimization. Notice that we draw just the matchable point $\mathbf{p}_m$ . . . . .	65
6.4	Synthetic experiments: normalized chi2 evolution. <code>opt</code> : starting from the optimal initial guess, <code>span</code> : initial guess from the spanning tree. . .	66
7.1	Semantic mapping system. . . . .	70
8.1	Comparison of the extraction procedure with the ROS <code>depthimage_to_laserscan</code> tool and our method in [101]. The RGB-D camera is represented by the black box on top of the robot, the virtual laser is represented by the 3D model of an Hokuyo sensor. . . . .	75
8.2	Semantic segmentation procedure: (a) input point cloud (b) models bounding boxes superimposed to the cloud (c) points belonging to the detected objects (each one is colored depending on the object type). . .	76
8.3	Semantic segmentation result: (a) input (b) output. Pixels that do not belong to an object are in black. . . . .	77
8.4	Representation of the <i>object-based</i> map. Each element has a: type (color), position (axes), size (bounding box) and model (point cloud). . .	78
8.5	Candidate pose computation. . . . .	81
8.6	Ray-casting procedure. The axes show the candidate view, gray lines represent the rays. The volumetric reconstruction is shown with wireframe cubes: occupied (red), green (free). . . . .	82
8.7	Simulation environment for experiments. . . . .	83
8.8	Example evaluation of the reconstructed map. Grey meshes are obtained from Gazebo models. Reconstructed objects are represented through: reference frame (position), blue box (size), colored point cloud (3D model). . . . .	83
8.9	Motion trajectories for both exploration strategies: (a) frontier-based, (b) object-based. Green dot: starting point. Red dot: end point. . .	86

# List of Tables

4.1	Comparison of metric representations. . . . .	42
5.1	Matchables table. The shape of $\mathbf{\Omega}_m$ discriminates the type of primitive represented by the matchable. The confidence ellipsoid obtained from $\mathbf{\Omega}_m$ is a sphere if the matchable is a point. If the primitive is a line or a plane the confidence ellipsoid degenerates respectively to a cylinder or to two parallel planes. . . . .	45
5.2	Information Matrix $\mathbf{\Omega}(\mathbf{m}, \mathbf{m}')$ for each possible pair of matchables.	47
5.3	ICL-NUIM - Relative Pose Error. . . . .	55
5.4	TUM - Relative Pose Error. . . . .	55
6.1	Noise figures for the second set of synthetic experiments. . . . .	64
6.2	TUM - Absolute Trajectory Error [m]. . . . .	65
7.1	Semantic Mapping Systems taxonomy. . . . .	72
8.1	Position and size errors for the reconstructed objects. . . . .	84
8.2	Comparison of position and size errors for the two exploration strategies.	87





# Chapter 1

## Introduction

Recent years witnessed relevant progress in different technologies and disciplines related to Robotics. More advanced sensing devices like the Velodyne Lidar and Microsoft Kinect allowed for more accurate environmental modeling (Fig. 1.1). Meanwhile, the huge amount of data available on the web and GPU computing enabled the development of Deep Neural Networks that showed impressive results in recognition tasks (Fig. 1.2).

These are enabling technologies that pave the way towards one of the long-standing goals of Robotics researchers: to have robots operating in human environments, by collaborating with humans or replacing them in demanding or dangerous activities. In this regard a fundamental question, that was also the inspiration for this research work, is: what does a robot need to perform the desired tasks in human environments?

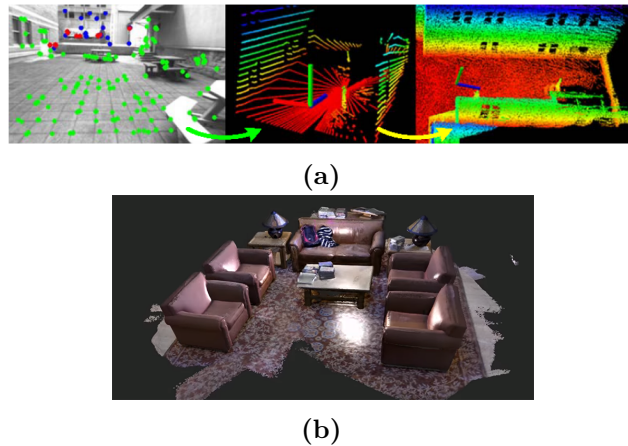
An introductory answer will be given in this chapter, whose structure is as follows: we first provide the reader with the research context (Sec. 1.1), we then highlight the issues addressed in this thesis (Sec. 1.2), finally we report the proposed contributions (Sec. 1.3).

### 1.1 Research Context

Robots, as we intend them today, have a quite recent history. They first appeared in sci-fi novels around the beginning of the twentieth century but just by the end of the fifties they started to be actually implemented.

Early models have been realized around the sixties and were the so-called manipulator arms: kinematic chains of *links* and *joints* that were employed in industrial plants for replacing humans in the assembly line. While, no more than ten years later, the first prototype of mobile robot was developed at the Stanford Research Institute.

These two types of robot present a substantial difference, the main reason being the environment in which they have to be deployed. Factory environments are designed for efficient production and no room for the unexpected is left. In this context, the task of a robot is defined by strict protocols. Conversely, man-made environments, *i.e.* all others but factories, are mostly unpredictable. This requires a mobile robot to perceive the situation around it and choose its actions accordingly.



**Figure 1.1.** Example of environment reconstructions obtained with: (a) Velodyne (image from [175]) and (b) Kinect (image from [171]).

Apart from this difference, there are some statements that are generally valid for robots. Every robot is designed to perform some tasks. Performing a task can be reduced to interacting with the surrounding environment by estimating its state and possibly modifying it. This interaction is practiced by executing a motion trajectory for the robot mobile base (mobile robot) or end-effector (manipulator arm).

To plan a motion trajectory, the robot requires a map. In this context, we refer to a map as a digital representation of the environment. Furthermore, basic operations on the map include inserting information when the robot acquires new knowledge and retrieving information when the robot has to perform a task.

For the majority of human environments such a map is not available or, if it is available, it can not be directly used. A typical example is the blueprint of a house. This contains an exact description of walls and doors but information about the furniture may be missing or outdated. For this reason, it is required to the robot the capability of processing sensors data to recover the information needed to operate in unknown environments.

In the next section we provide an introduction to this problem, present a quick overview on the current development and highlight the particular aspects addressed in this thesis.

## 1.2 Addressed issues

Building a digital representation of the environment is a well-known problem in the Robotics community and it is typically referred to as Simultaneous Localization and Mapping (SLAM). Solving the SLAM problem usually consists in finding the configuration of sensor poses and landmark positions that best fit the sensor measurements.

For its relevance, this problem has attracted a lot of attention in the last three decades and a considerable amount of literature has been produced. At the core of modern SLAM systems there are methods to register sets of data expressed in different reference frames, which we survey in Sec. 2.1. Additionally, *graph-based*



parameterizations.

This allows to believe that a mapping method to obtain both robustness and usability is still not available. Consequently, it is a common practice to address these two requirements separately. As reported in Sec. 2.3, the problem of building a model of the environment that enables a more advanced collaboration between humans and robots goes under the name of Semantic Mapping.

This can be basically addressed by enriching the representation obtained from SLAM with human concepts like objects and places. Despite different modalities exist for this purpose, like *speech understanding* and *information retrieval*, *scene analysis* and *image understanding* techniques are taking over. In Sec. 2.3.1 we present the state-of-the-art.

Beyond that, an important insight gained from the literature review is that for generating usable semantic maps there are also two other aspects that can not be neglected. On one side, it is not sufficient to attach semantic labels to the SLAM map entities. This is because, as explained earlier, they are too low-level. Instead, geometric and semantic information, coming respectively from SLAM and image analysis techniques, should be combined to build an *ad-hoc* representation (Sec. 2.3.2). On the other side, to obtain accurate models of the objects, that the robot has to interact with, automated strategies for planning sensor motions must be devised (Sec. 8.1.3).

### 1.3 Contributions

According to the issues mentioned in the previous section, in this thesis we present the following contributions:

- a unified representation for a 3D scene consisting of the following geometric primitives: points, lines and planes;
- both a *local registration* and a *global optimization* algorithm that exploit this representation for robust estimation;
- the architecture of a *semantic mapping* system capable of building an *object-based* map with no *prior* information about the environment whose feasibility is demonstrated in simulation;
- a pedagogical instructive presentation of the SLAM problem and factor graphs.

Part of the content of this thesis is the result of papers published in international journals and conferences. In the remainder of this section these publications are listed chronologically:

- [F. NARDI](#) AND [M.T. LÁZARO](#) AND [L. IOCCHI](#) AND [G. GRISSETTI](#), “**Generation of laser-quality 2d navigation maps from rgb-d sensors**”, In *RoboCup Symposium*, 2018
- [F. NARDI](#) AND [B. DELLA CORTE](#) AND [G. GRISSETTI](#), “**Unified Representation and Registration of Heterogeneous Sets of Geometric Primitives**”, In *IEEE Robotics and Automation Letters (RA-L)*, 2019

- 
- I. ALOISE AND B. DELLA CORTE AND F. NARDI AND G. GRISETTI, “**Global Optimization Using Heterogeneous Geometric Primitives**”, In *IEEE Robotics and Automation Letters (RA- L)*(submitted, in revision), 2019



Part I  
Basics





## Chapter 2

# Related Work

This chapter is devoted to a literature survey for presenting relevant works that focused on the problems introduced in Chapter 1. We first provide an overview on Geometric Registration in Sec. 2.1, which is a fundamental technique to perform mapping. Subsequently, in Sec. 2.2, we review the state-of-the-art SLAM algorithms, represented by graph-based approaches, as well as methods that extend their formulation with geometric primitives other than points. Finally, in Sec. 2.3 we focused on Semantic Mapping. Here, we considered separately the various aspects involved in solving this problem: extracting semantic information from visual data, building a semantic representation of the environment and planning sensor motion to improve the acquired knowledge.

### 2.1 Geometric Registration

Geometric registration consists in seeking for an alignment between sets of data expressed in different reference frames. It is an ubiquitous problem in Computer Science and Applied Mathematics research fields. For this reason, it has been extensively investigated in Computer Graphics [127, 159, 152] for *surface reconstruction* and *shape matching*, as well as in Robotics [117, 5], where it is mostly applied for *mapping* and *localization*. Then, depending on the application, this information can be used to measure how a set fit into another (shape matching), to estimate a relative pose (localization) or to integrate the sets in a common reference frame (reconstruction).

Registration approaches can be divided in two general classes: global methods, that match transform-invariant descriptors and local methods, that rely on an initial guess of the alignment.

#### 2.1.1 Global Registration

Global registration methods share a common pipeline. Given two sets of geometric primitives (usually points), called *model* and *data*, the first step is to sample interesting points and compute a geometric descriptor [128, 143] for each of them. After that, a matching procedure is performed to find correspondences between the two sets. These matches can then be used to find the transform that better aligns

them with error minimization. Since rigid transforms can be entirely determined by only 3 point matches, a possible solution is to adopt the Random Sample Consensus (RANSAC) scheme [40]. However, despite its simplicity, this method has a non neglectable computational complexity that, in case of 3D points, is cubic in the number of points [68, 16].

Gelfand *et al.* [49] proposed a more efficient scheme by exploiting robust shape descriptors for feature identification before alignment and branch-and-bound optimization. Li and Guskov [88] focused their work on the sampling problem and introduce a multi-scale salient feature extraction algorithm, inspired by SIFT features [92]. Aiger *et al.* [1] reduced the number of trials required to establish a reliable registration by extracting all coplanar 4-points sets that are approximately congruent, under rigid transformation, to another given set of coplanar 4-points. This brings the computational complexity from cubic to quadratic.

Papazov and Burschka [115] attempted to obtain a global optimal alignment by explicitly tackling noise and outliers in input data. To this end, they casted a non-linear stochastic optimization problem and used Simulated Annealing (SA) [75] to solve it. Their choice is motivated by the fact that SA algorithms manage to explore regions around points in search space at which the objective function takes values greater than the current minimum, enabling it to escape from local minima. Feature matching is notably addressed in the work of Cheng *et al.* [19]. Here the idea is that pointwise matching can fail in the presence of ambiguities such as repeated elements or similar local appearance. The authors instead propose matching groups of features by using super-symmetric tensors to represent the constraints between the tuples.

A different perspective on the problem has been proposed by Rodola *et al.* [123] who base the registration methodology on a game-theoretic inlier selection approach [99]. In this framework, the matching problem is turned into the selection of a small group of pairs, whose coherence follows from the notion of compatibility.

### 2.1.2 Local Registration

All methods mentioned so far rely on the assumption that matching features between the *data* and *model* sets can be solved independently from the initial alignment. This is convenient when this information is missing or the two sets display large displacement. However these situations may involve high computational cost or sub-optimal estimation. When an initial guess is available or the two datasets are acquired at nearby locations, a better solution is to employ local registration methods.

In this context, a well known solution is the Iterative Closest Point (ICP) algorithm [8]. Despite its spread, the vanilla formulation of the method presents two main drawbacks, namely: the assumption of perfect correspondence between the *data* and *model* sets and the requirement of uniform sampling from the underlying surface. To overcome these limitations, many solutions have been proposed [127].

Zhang [177] eliminated the perfect correspondence constraint by adding a robust method of outlier rejection in the correspondence selection phase of the algorithm. This allows to register scans acquired in realistic conditions. Chen and Medioni [18] addressed the problem of object reconstruction from range images. Their method is

based on the idea that most range data is sampled from “locally planar” surface, thus indicating the *point-to-plane* distance as a more meaningful metric and eliminating the need for the uniform sampling assumption. Segal *et al.* [135] pushed this approach a step further by proposing a probabilistic formulation of the problem that includes [18] and, in general, allows to model sensor uncertainty in the error minimization phase.

Because of its simplicity, the robotics community has vastly adopted the ICP algorithm as a core component for solving the SLAM problem. Lu and Milos [94] demonstrated the effectiveness of the method for a wheeled mobile robot equipped with a SICK rangefinder. Some years later, thanks to the advance in laser sensing technologies, robotics researchers managed to employ registration techniques in a growing number of applications. As an example, Nüchter *et al.* [108] presented a 6D SLAM system capable of recovering the 3D geometry of the environment and estimating the 6DOF pose of the robot. Pathak *et al.* [116] propose a direct method to solve the local registration problem based on the extraction of planar patches from unorganized 3D point clouds. Contrary to other methods derived from ICP, they perform least-squares pose estimation in one step by assuming that the uncertainty associated to plane normals is directionally uniform. This leads to the decoupling of the plane parameter covariance matrix. As a consequence, this method guarantees fast registration of large scenes even with limited computational resources. However, it is mainly tailored for *stop-and-go* fashion mapping scenarios where the range sensor is mounted on a rotating station and a single frame captures a large portion of the environment.

In the last decade, with the introduction of affordable RGB-D cameras, a new wave of investigation hit the community [59, 118]. Despite their precision is not comparable with that of laser technology, RGB-D cameras provide rich visual and depth information, making them very convenient for digital reconstruction. Steinbrücker *et al.* [144] presented Dense Visual Odometry (DVO), an energy minimization approach for visual odometry that directly exploits both the intensity and depth information. The idea is that, for each edge in the intensity channel, can be computed the 3D location from the depth channel. Then, the transformation is found by minimizing the reprojection distance of the edges on the next frame. The main advantage is that, because of the few processing required, this method can be executed at high frame rates, making true the assumption of small displacement between frames. On the other hand, it is very sensitive to intensity blur and severe lighting conditions.

Another relevant approach is KinectFusion (KinFu), proposed by Newcombe *et al.* [103]. This work employs the implicit representation of surfaces to model the scene geometry. More in detail, they fuse sensor readings in a Truncated Signed Distance Function (TSDF) [26], to obtain a very detailed 3D model of the environment. A TSDF is a scalar field, defined on a 3D grid, that stores for each cell a signed distance to the underlying surface. The quality of the representation comes at high computational cost. Indeed, to obtain real-time performance the authors make use of GPU parallelization and the construction and management of a 3D grid greatly reduces the scalability of the method. After the original method, several variants have been proposed to tackle these issues [169, 171].

Recently, Serafin and Grisetti [137] presented the Normal Iterative Closest Point (NICP) algorithm, a variant of ICP that can run on-line on a multi-core CPU. The

novelty of this method consists in using a 6D error metric that extends the original error metric of ICP by considering not only the distance between corresponding points but also between corresponding surface normals. Additionally, the method uses the surface curvature as additional cue for data association.

Even before the advent of RGB-D cameras, Magnusson and Duckett [97] introduced the 3D Normal Distribution Transform (NDT), a volumetric representation for the model surface that describes the probability of finding a point at a certain position. One of the advantages of this representation is that it simplifies the step of data-association, but, as for the TSDF, the descriptivity of the model is bounded to the grid resolution.

### 2.1.3 Geometric Primitives Detection

All methods presented so far attempt to register *low-level* geometric primitives, *i.e.*, points or voxels. This is mainly due to the existence of well-known techniques for detecting and matching such primitives. In contrast to that, another line of research, which is what we follow, has gone in the direction of augmenting these techniques with *high-level* features like lines and planes. In the remainder of this section, we review the literature regarding the detection of such primitives both from images and point clouds.

#### Points

Point feature detection is a fundamental step for solving problems like *object recognition* or *3D reconstruction*. For RGB images a vast literature on this problem exists. Proposed methods can be divided in: edge-, corner- and blob-based approaches.

Edges are sets of pixels where the image intensity changes neatly and, consequently, the gradient has high magnitude. This usually happens at the frontiers between distinguishable regions. Popular edge detection methods are based on differential operator techniques and high-pass filtering [46]. A more robust approach has been proposed by Canny [13], who defined a method to guarantee low error rate, good localization and minimal response.

A pixel in which two edges intersect or, equivalently, whose neighborhood has two or more non-parallel edges is a corner. Additionally, since corners can be detected through the image gradient curvature, in some cases, also a small white spot on black background is considered to be a corner. Most feature detection methods compute a corner response function for each image pixel. Pixels which exceed a threshold cornerness value (and are locally maximal) are then retained [57, 154]. Despite it is possible to obtain high quality features with these methods, they do not meet real-time requirements. Another category of corner detectors work by comparing a small patch of an image with a “template” corner, since they do not involve second derivatives computations, noise reduction is not needed. This greatly improves timing performance, allowing these methods to be used in most computer vision and robotics applications [124].

A blob is formed by a group of connected pixels that share analogous characteristics. Many interest point detection methods work by detecting corners at different scales, thus, in general, the term blob captures also the concept of interest

point. Blob-based approaches aim at identifying the unique regions in an image by comparing local properties (e.g., intensity and colour) to their neighboring regions in scale spaces. The Scale Invariant Feature Transform (SIFT) algorithm by Lowe [92] is considered one of the most relevant. This method uses a pyramidal approach to make the features invariant to the scale and assigns them orientation to obtain invariance to image rotation.

### Lines

Line segments can be defined as intervals of a straight line in an image that are orthogonal to the image gradient at most of their points [165].

A seminal work in the detection of line segments is the Hough Transform Method (HTM) [67]. This method detects line segments in an image through a voting scheme. That is, if for a certain point of the image there is a family of lines that can pass through it, we can say, alternatively, that a line is “voted” by a certain number of points. The Hough method works by counting for each line the number of votes received by edge pixels in the image. A line is detected when the votes exceed a certain threshold.

As pointed out by Desolneu *et al.* in [32], a drawback of this method is that it supposes that what it wants to find, in this case lines, is in the image. Building on this idea, Von Gioi *et al.* [166] presented the Line Segment Detector (LSD), a parameterless method, that can be considered state-of-the-art. It computes for each pixel a *level-line* that is the direction of the intensity gradient at that pixel. Then, a clustering procedure detects the so-called line support regions and a validation procedure retains only good line candidates. This method is aimed at solving three fundamental issues: over-detection (false positives), under-detection (false negatives) and the accuracy of each detection.

### Planes

All techniques mentioned so far detect features in intensity images. On the other hand, plane extraction works directly on 3D data. This problem received great interest in the past years, since it is a fundamental component of many scene understanding and reconstruction applications. Proposed methods in literature mainly fall in two classes: RANSAC-based and region-growing approaches.

RANSAC-based methods work by randomly selecting a plane model from a subset of the input data and then comparing it to the remaining points, once a plane model is supported by a sufficient number of points, they are removed from the input data and the procedure continues until no other planes can be found. A clear drawback of this approach is that uninformed selection can increase computation time [134].

Oehler *et al.* [109] combine Hough transform with RANSAC to improve efficiency. Their plane model selection is based on a coarse-to-fine strategy: they compute surface normals at different resolutions and discard surfels whose normal is inconsistent with planes at a coarser resolution. Remaining surfels are fed to the Hough transform to perform a rough segmentation of the scene and then RANSAC is used for robust detection. As a final step, co-planar connected plane segments are merged.

Similarly, Hulik *et al.* [66] apply RANSAC on local patches of the point cloud, then use a seed-fill algorithm to find all points in the patch that fit the plane and then grow the region from the locally found plane instance to the whole point cloud.

On the other hand, region-growing methods work by selecting random points and then expanding a region around it until a certain criterion is satisfied. Poppinga *et al.* [119] grow a region by considering each time a point that is not further than a certain distance. Then, the selected point is added to the region if the mean square error (MSE) to the plane of the region is less than a threshold. Holz *et al.* [62] extend this approach by adding as a pre-processing step the estimation of surface normals, and incrementally update the plane's normal equation. This has the effect of improving time performance.

RGB-D cameras provide organized point clouds that are particularly suited for 3D segmentation and structure detection. Feng *et al.* [39] construct a graph by dividing the input cloud into non-overlapping regions according to point neighborhood information. Consecutively, agglomerative hierarchical clustering (AHC) is performed on the graph. Planes are detected by finding the region that has the minimum fitting MSE and merging it with the neighbor that minimizes MSE.

Trevor *et al.* [157] propose an efficient method for segmenting organized point clouds based on connected-component segmentation. In addition to color information, they segment image regions exploiting also 3D coordinates of each point and surface normals.

## 2.2 Simultaneous Localization and Mapping

In the last three decades, the SLAM problem gained considerable attention from the Robotics community and two main approaches have been proposed: filtering and smoothing. Filtering approaches are usually defined as on-line SLAM methods, since they aim at estimating the current robot pose and the map as a new measurement becomes available. On the other side, smoothing approaches estimate the full trajectory of the robot and the map from all the measurements acquired by the robot. Therefore, are said to solve the full SLAM problem.

In the remainder, we will focus on relevant approaches to solve the latter. In this context, the optimization problem is usually modeled using a factor graph [52]. Therefore we refer to graph-based techniques. Furthermore, we will present methods that consider the usage of different geometric primitives in the SLAM formulation.

### 2.2.1 Graph-based SLAM

Lu and Milios [93] laid the foundation of graph-based approaches. Their method derives from the necessity of handling inconsistencies that may arise in the mapping process. This typically happens when the robot revisits a place after some time and, due to error accumulation, there is no overlap between the past and the current pose. To handle this situation, they propose to build a network of spatial relations between robot poses and landmark observations, so that, when an inconsistency is detected, it is possible to propagate the corrections to all the related poses.

Leveraging on the same idea, Gutmann and Konolige [54] extended this work with the Local Registration and Global Correlation (LRGC) method to obtain robust

loop detection and efficient optimization. Their method works by accumulating consecutive scans in *local maps*. This allows: (a) to find the correlation between two poses even if they have large initial displacement and (b) to perform global optimization only upon loop detection instead of doing it for each new measurement.

Since then, many approaches for minimizing the error in the constraint network have been proposed. Howard *et al.* [65] generalized the SLAM problem by representing it on a mesh of springs and proposed a relaxation scheme to solve it. In a similar fashion, Duckett *et al.* [34] proposed a fast, on-line energy minimization algorithm. However their convergence rate is linear, involving more iterations compared to Least Squares. A refined version has been presented by Frese *et al.* [43] that employs multi-level relaxation to increase the convergence speed.

The complexity of solving the SLAM problem can be mainly associated to: (1) the large search space deriving from the high number of robot poses and landmarks, (2) the bad initial guess obtained from odometry. Olson *et al.* [112] introduced a nonlinear map optimization algorithm aimed at dealing with these issues. In particular, they proposed an alternative state space representation that allows to update many poses in a single iteration and a variant of Stochastic Gradient Descent (SGD) that is robust to local minima.

Grisetti *et al.* [53] extended this approach by introducing a tree-based network parameterization, so that the complexity of the solution depends only on the size of the environment and not on the robot trajectory. Subsequently, Ni *et al.* [104] and Grisetti *et al.* [51] applied divide and conquer strategies to address the issues of poor initial guess and scalability. While the first method relies on nested dissection to solve the linear system, the latter assembles a set of non-linear sparser problems from local portions of the graph.

A relevant aspect of the SLAM problem is that in a single constraint only a small subset of the state variables are involved. This results in a sparse Hessian matrix of the linear system. Dellaert *et al.* [31] exploited this sparsity in their system, known as  $\sqrt{SAM}$ . Afterwards, Kaess *et al.* presented an on-line version with partial reorderings [72], to compute the sparse factorization, and new data structures to the original system configuration [71].

In parallel Kümmerle *et al.* [83] proposed  $g^2o$ , an optimization tool that allows to easily prototype sparse least-squares solvers for factor graphs. This is designed in a way to separate the problem definition from its solution and can be easily extended through plugins. Enabling the user to apply different strategies to solve the factor graph, and to extend the types of factors and node variables.

### 2.2.2 SLAM with Geometric Primitives

The straightforward benefit of using high-level features has been extensively exploited to improve the robustness of landmark-based SLAM approaches. Castellanos *et al.* [14] proposed the symmetries and perturbation map (SPmap). In their model, a geometric feature is represented by combining a perturbation vector that expresses the uncertainty about its location and a binding matrix that accounts for its shape. Despite providing a powerful tool to represent and propagate uncertainty for any type of geometric entity, the authors limit themselves to line segments.

Building on this idea, other approaches have been presented that aim at improving

the accuracy of camera tracking with edges and lines. Klein and Murray [77] augment the map with edge features and rely on their resilience to motion blur for obtaining more robust tracking in case of rapid camera motions. Eade and Drummond [35] follow the same paradigm and extend it by paying particular attention to the selection, observation and estimation problem of edge features. In both works edges are parametrized as points with a direction. Lemaire and Lacroix [87] presented an effective method to incorporate lines in monocular SLAM. In particular, they propose a *delayed* landmark initialization step based on Gaussian sum. This consists in adding the landmark to the map only after a certain number of observations are gathered, then each new observation is used to update the probability distribution that will eventually converge to a single Gaussian. The authors make use of Plücker coordinates to represent 3D line segments. Smith *et al.* [141] integrated straight lines into a monocular Extended Kalman Filter SLAM (EKF SLAM) system without compromising timing performance. In particular, they propose a simplistic yet effective method to detect line segments in the image frame. Instead of fitting methods or Hough transform, which are not suited for online operation, they search for corners in the image and then evaluate all possible lines between them. As a consequence, lines are parametrized through their endpoints.

As mentioned earlier, the advent of RGB-D cameras implied a greater diffusion of mapping techniques and SLAM with high-level features makes no exception. Choi *et al.* [20] exploit rich information provided by these sensors, in terms of intensity and depth, through the detection of various types of edges. Apart from edges in the color channel (Sec. 2.1.3), they also extract edges by considering depth and normals discontinuities.

Lu and Song [95] proposed a robust RGB-D visual odometry approach by combining points and lines. Points are extracted with [4] and back-projected in 3D leveraging on depth information, while 3D lines are detected directly from the depth with a RANSAC method.

Additionally, organized (and colored) point clouds returned by these sensors are particularly suited for 3D segmentation and structure detection and raised interest in registration methods that consider planes as higher level features [70, 158, 151, 96].

## 2.3 Semantic Mapping

Acquisition and modeling of semantic information is a key requisite for mobile robots to be deployed in human environments. In this field, fundamental aspects faced by research are: the recognition of places and objects, the construction of semantic models and the exploration strategies to enrich contextual knowledge.

In this section, we will present relevant work that focused on the mentioned problems, namely: semantic information extraction, map construction and active vision.

### 2.3.1 Semantic Information Extraction

Thanks to relevant advances in Computer Vision, a growing number of robotics applications are designed to extract semantic information from images. Image Analysis can be decomposed into sub-tasks, depending on the information one is



interested to extract from the input data. These sub-tasks can be organized on a progression that goes from coarse to fine grained inference.

Image classification is the task of assigning a semantic label to an input image from a fixed set of categories. Ulrich and Nourbakhsh [160] propose an appearance-based place recognition system for topological localization. They use colour histogram features [148] and a simple voting scheme for nearest-neighbor matching. In a similar fashion, Torralba *et al.* [156] derive an Hidden Markov model (HMM) for place recognition and new place categorisation based on the global statistic feature retrieved from texture [111]. In contrast, Lisin *et al.* [89] propose to model classes of images as a probability distribution over local features, in order to be combined with global features. This method has proven to perform well in applications where a rough segmentation of objects is available.

Object detection consists in making a prediction not only of object categories but also of their spatial locations. A seminal work can be considered that of Viola and Jones [164], who proposed a fast and robust face detection. Their method makes use of Haar-like features [114] to search for likely face candidates, which can then be refined using a cascade of more expensive but selective detection algorithms [44]. Likewise, a well-known example of pedestrian detection has been proposed by Dalal and Triggs [27], who use a set of overlapping Histogram of Oriented Gradients (HOG) descriptors fed into a Support Vector Machine (SVM) [24].

Image segmentation is the task of finding groups of pixels that possess some “similarity” and is one of the oldest and most widely studied problems in Computer Vision. Early techniques focus on local region merging and splitting [110, 10], while, more recent algorithms often optimize some global criterion, such as intra-region consistency and inter-region boundary lengths or dissimilarity [23, 139, 38, 15, 113].

Despite the popularity of the presented methods, a recent breakthrough in Scene Understanding has been the adoption of Convolutional Neural Networks (CNNs) [47]. Krizhevsky *et al.* [81] presented the pioneering deep CNN that, despite its simplicity, won the Imagenet 2012 classification challenge with wide margin on the closest competitor. Similarly, different object detection methods based on deep neural networks have shown to outperform the state-of-the-art [122, 37, 90]. Consequently, the capabilities of such networks have been also investigated in pixel-level labeling problems like semantic segmentation. In this context, a milestone is the work of Long *et al.* [91] who transformed existing classification models ([140, 149]) into fully convolutional ones to output spatial maps instead of classification scores. One of the main reason behind its popularity is that, with this approach, CNNs can be trained end-to-end and efficiently learn to make dense predictions with inputs of arbitrary size.

### 2.3.2 Map Construction

To enable the robot to perform complex tasks one is faced with the problem of augmenting metric maps with human-level knowledge, like: object/place categories, functions, properties and so on. A traditional view of the problem is that of overlapping to the metric map other representation layers to encode the information needed by the robot to perform its tasks [45].

Zender *et al.* [173] proposed a system to build a Multi-Layered Spatial Rep-

resentation with different modalities, composed of three modules: the perception subsystem for evaluation of sensors input, the communication subsystem for situated spoken dialog and the subsystem for multi-layered conceptual spatial mapping that grounds the semantic symbols to the map features. As a result, their representation of the environment adds to the metric map a navigation and a topological map, with the nodes of the latter linked to symbols in a conceptual map that can be exploited by the robot for reasoning. In a similar fashion, Pronobis and Jensfelt [121] presented a probabilistic framework to fuse information coming from heterogeneous modalities into their layered representation.

With the progress in visual and ranging technologies there has been a growing interest in extracting semantic information directly from images or 3D point clouds. In this context, the route has been traced by the work of Nüchter and Hertzberg [107]. Their processing pipeline is composed as follows: they first obtain a metric reconstruction of the environment with a wheeled robot equipped with a 3D laser scanner; then, a first interpretation of the scene is performed by classifying regions of the final point cloud as belonging to floor, walls, ceiling and so on; finally, a shape matching procedure is used for object detection. Despite its novelty, this method was able to work only under strong assumptions.

Leveraging on this idea, Rusu *et al.* [130] proposed a region growing segmentation technique based on surface geometric information, *i.e.*, normals and curvature. Then, since their robot was assumed to operate in kitchen environments, the object detection routine was performed with cuboid fitting. Tenorth *et al.* [153] build their environment representation, called KNOWROB-MAP, with a similar pipeline and focus particularly on the symbol grounding aspect [56]. To do so, they extend the recognition system with encyclopedic, action-related and common-sense knowledge. To perform segmentation of general indoor scenes, Koppula *et al.* [79] proposed a learning approach. Their method is based on a Probabilistic Graphical Model (PGM) whose classification features are: visual appearance, local shape and geometry and geometrical context. Similarly, Valentin *et al.* [161] proposed to perform Surface Reconstruction for obtaining a surface mesh and, on that, use a Conditional Random Field (CRF) [84] for semantic inference.

These methods are limited by two main factors: (a) they need a reconstruction of the environment for objects/places recognition, (b) they return a coarse segmentation of the scene. At the same time, impressive image understanding results have been obtained with Random Decision Forests (RF) [9] and Convolutional Neural Networks (CNNs) [82]. For this reason, thanks to the availability of affordable RGB-D cameras, a new paradigm for Semantic Mapping arised that combines reconstruction from range information and recognition from visual information.

Stüeckler *et al.* [145] proposed an on-line method following this paradigm. They use SLAM to register different views of the scene in a common reference frame and build a volumetric reconstruction. Subsequently, they use RFs to obtain object-class segmentation of the views and fuse per-voxel predictions with a Bayesian filter. Following this approach, Hermans *et al.* [60] proposed a dense 3D Semantic Mapping system. Unlike [145], they use Random Forests to obtain per-pixel (dense) classification. Additionally, after Bayesian fusion, prediction results are smoothed with a CRF.

Similarly, Vineet *et al.* [163] applied the same technique on data acquired with a

stereo camera, to extend this method to outdoor and large-scale environments, and showed remarkable results on the well-known KITTI dataset [48]. Finally, a step further has been made by McCormac *et al.* [98] with SemanticFusion, where they have been able to obtain improved semantic segmentation thanks to recent advances in Convolutional Neural Networks.

The success of several classification methods may be credited to environment specific training. To alleviate this need, Sünderhauf *et al.* [147] proposed a place categorization method for mobile robots that is *transferable* and *extendible*. These features are obtained using CNNs, that generalize well, combined with *one-vs-all* classifiers, that can learn to recognize new classes online. As a consequence, this method is particularly suited for long-term autonomous operations and life-long learning.

Brucker *et al.* [11] propose a novel place categorization method based on semantic information acquisition from different modalities. The input to their method is a 3D textured mesh obtained with Surface Reconstruction techniques. This is projected on a 2D plane at the floor level to obtain an *occupancy grid* representation of the environment. The obtained map is first segmented based on geometric cues and then processed to sample a set of viewpoints for rendering RGB-D frames from the mesh representation. Finally, classification results coming from both sources of information are fused together employing a CRF.

Semantic maps typically model facts about robot percepts as either true or not, neglecting uncertainty arising, for example, from the inaccurate modeling of elements in the robot workspace. Ruiz-Sarmiento *et al.* [126] tackle this issue by employing a Probabilistic Graphical Model, namely a CRF, that allows to measure beliefs about symbol grounding and perform probabilistic inference according to spatial relations among percepts. To this end, they present a novel representation, called Multiversal Semantic Map (MvSmap), that considers different possible groundings as instances of ontologies with probability values for their grounded concepts and relations.

### 2.3.3 Active Vision

To support complex task execution, it is necessary to provide the robot with the ability of actively building object models. Foissotte *et al.* [41] consider the problem of building 3D models of an object for a humanoid robot equipped with a dense range sensor. To this end, they limit the sensor configuration space to a sphere around the object and cast the NBV evaluation into an optimization problem. The object is modeled with a 3D occupancy grid, thus, the objective function is maximized according to the amount of *unknown* visible voxels.

On the other hand, Torabi and Gupta [155] presented an integrated eye-in-hand system for 3D object modeling. The main novelty of this work consists in the fact that the NBV planning is computed in the 6DoF configuration space of the sensor that is not known *a-priori*. This allows this method to be employed in unknown environments and in situations where the objects to be modeled can not be moved. However, the high dimensionality of the search space and the inverse kinematics computation render this method computationally expensive.

Alternatively, Potthast and Sukhatme [120] proposed to plan in Cartesian space and presented an efficient approach based on Probabilistic Road Map (PRM) planners.

Vazquez-Gomez *et al.* [162], instead, impose a pre-defined set of views for each object to reconstruct and define an accurate NBV evaluation metric based on: information gain, collision avoidance, surface overlap and path cost.

So far, the presented methods addressed only the geometry reconstruction problem. Eidenberger and Scharinger [36] propose a sensor planning method based on Partially Observable Markov Decision Processes (POMDPs) to model and recognize all objects in a scene. The predicted effects of future sensing actions are measured through an information gain metric and the next best sensing action is determined to improve the scene knowledge. The main drawback of this method is that a reward function needs to be learned beforehand.

Stampfer *et al.* [142] proposed a method that exploits active vision to improve object recognition. Assuming to have *a-priori* knowledge of the objects present in the scene, they adopt an *information-driven* view planning approach. This has the advantage of allowing to fuse the outcomes of different recognition routines but, at the same time, the number of objects for which this method is applicable is limited.

Kriegel *et al.* [80] proposed an integrated system for active scene exploration intended at both reconstructing and recognizing objects in the scene. They used a robot equipped with both an accurate 3D laser scanner and an RGB-D camera. The former is used to build a models database for the detected objects during the exploration of the scene, while the latter serves to determine the next best view.

Wu *et al.* [172] tackled the 3D object recognition and pose estimation for indoor table top scenes. They propose an active recognition strategy suitable for dynamically exploring a cluttered scene while localising the pre-trained objects. In their approach, an object is represented using two models: a dense point cloud model, used for 3D Object Recognition and a sparse feature cloud model, used for view evaluation.

## Chapter 3

# Fundamentals

### 3.1 State Estimation

Mathematical optimization is a powerful tool that can be used to solve a variety of problems in applied sciences. In many, if not all, engineering disciplines optimization techniques allow to solve real-world problems in effective and elegant ways. Computer and robotics engineers make no exception, since they heavily employ Least Squares methods to solve state estimation problems, particular examples are registration and SLAM.

#### 3.1.1 Bayesian Framework

In many application scenarios the state of a system can be described by a set of variables  $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ . Usually, these variables cannot be directly measured, rather, we observe indirectly the state of the system through a set of measurements  $\mathbf{z} = \{\mathbf{z}_1, \dots, \mathbf{z}_K\}$ . In real world, measurements are corrupted by noise, thus, each  $z_k$  is a random variable and estimating the state of the system turns into computing the probability distribution over the possible states given the measurements. Formally, the goal is to estimate:

$$p(\mathbf{x} \mid \mathbf{z}) = p(\mathbf{x}_1, \dots, \mathbf{x}_N \mid \mathbf{z}_1, \dots, \mathbf{z}_K) \quad (3.1)$$

$$= p(\mathbf{x}_{1:N} \mid \mathbf{z}_{1:K}) \quad (3.2)$$

In practice there is no closed-form solution to obtain Eq. (3.2). However, Bayes rule, along with simplifying assumptions, can be adopted to turn the problem in a

solvable form:

$$p(\mathbf{x}_{1:N} | \mathbf{z}_{1:K}) = \quad (3.3)$$

$$[\text{Bayes rule}] = \frac{\overbrace{p(\mathbf{z}_{1:K} | \mathbf{x}_{1:N})}^{\text{likelihood}} \cdot \overbrace{p(\mathbf{x}_{1:N})}^{\text{prior}}}{\underbrace{p(\mathbf{z}_{1:K})}_{\text{normalizer}}} \quad (3.4)$$

$$[\text{uniform prior}] = \frac{p(\mathbf{z}_{1:K} | \mathbf{x}_{1:N}) \cdot p_x}{p_z} \quad (3.5)$$

$$= \eta p_x p(\mathbf{z}_{1:K} | \mathbf{x}_{1:N}) \quad (3.6)$$

$$[\text{measurements independence}] \propto \prod_k p(\mathbf{z}_k | \mathbf{x}_{1:N}). \quad (3.7)$$

Bayes rule allows to decompose our initial distribution in three terms. The *likelihood* or *observation model*  $p(\mathbf{z}_k | \mathbf{x})$  encodes the probability of obtaining the measurement  $\mathbf{z}_k$  in case the system is in the state  $\mathbf{x}$ . The *prior*  $p(\mathbf{x}_{1:N})$  assigns a probability to each possible state  $\mathbf{x}_{1:N}$ , assuming we have no *a priori* knowledge of our system, this is a constant. Finally, considering that the measurements are conditionally independent from the state, the joint distribution is equivalent to the product of the marginals and we get to Eq. (3.7). This equation tells us that the initial distribution that we were looking for is proportional to the *likelihood*, therefore, we can turn the original problem the other way round and search for the state that better explains the measurements or, in other words, that maximizes their likelihood.

### 3.1.2 Gauss-Newton

The measurements *likelihood* depends on the particular sensor used. Assuming that we have a mathematical model that describes the behavior of our sensor and that the noise affecting the measurements is zero mean normally distributed, we can choose to model the likelihood as:

$$p(\mathbf{z}_k | \mathbf{x}) \propto \exp(-(\hat{\mathbf{z}}_k - \mathbf{z}_k)^\top) \quad (3.8)$$

where  $\hat{\mathbf{z}}_k = \mathbf{h}_k(\mathbf{x})$  is the predicted measurement obtained from the sensor model and  $\mathbf{\Omega}_k = \mathbf{\Sigma}_k^{-1}$  is the information matrix of the conditional measurement.

However, we notice that in most situations the sensor model  $\mathbf{h}_k$  is a non-linear function of the state. Therefore, even if the measurement noise is gaussian,  $p(\mathbf{z}_k | \mathbf{x})$  will not be normally distributed. To fix this, we can apply Taylor expansion to obtain a linear approximation of the sensor model,

$$\mathbf{h}_k(\check{\mathbf{x}} + \Delta \mathbf{x}) \simeq \underbrace{\mathbf{h}_k(\check{\mathbf{x}})}_{\check{\mathbf{z}}_k} + \underbrace{\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\check{\mathbf{x}}}}_{\mathbf{J}_k} \cdot \Delta \mathbf{x}, \quad (3.9)$$

and, consequently, a gaussian likelihood.

Now, let us rewrite Eq. (3.8). In particular, we decide to linearize  $\mathbf{h}_k$  around the optimal state  $\mathbf{x}^*$ , that is, the state that better fits the measurements:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{z} | \mathbf{x}). \quad (3.10)$$

In this way, by substituting  $\mathbf{h}_k(\mathbf{x}^* + \Delta\mathbf{x})$  in Eq. (3.8) and treating  $\mathbf{x}^*$  as constant, we obtain a new distribution  $p(\mathbf{z} | \Delta\mathbf{x})$  that depends only on the increments  $\Delta\mathbf{x}$ :

$$p(\mathbf{z}_k | \mathbf{x}^* + \Delta\mathbf{x}) \propto \exp \left[ -(\mathbf{h}_k(\mathbf{x}^*) + \mathbf{J}_k \Delta\mathbf{x} - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\mathbf{x}^*) + \mathbf{J}_k \Delta\mathbf{x} - \mathbf{z}_k) \right]. \quad (3.11)$$

Eq. (3.11) measures how the likelihood changes according to a small perturbation of the state. Having a gaussian likelihood allows us to formulate an optimization problem with a closed-form expression and, letting the gaussian depend on the state increments, to find a solution to the said optimization problem.

The optimization problem can be formulated as follows. We want to find the optimal state  $\mathbf{x}^*$  that maximizes Eq. (3.8):

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{k=1}^K p(\mathbf{z}_k | \mathbf{x}) \quad (3.12)$$

$$[\text{gaussian assumption}] = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{k=1}^K \exp[-(\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)] \quad (3.13)$$

$$[\text{taking the logarithm}] = \underset{\mathbf{x}}{\operatorname{argmax}} \sum_{k=1}^K [-(\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)] \quad (3.14)$$

$$[\text{removing the minus}] = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{k=1}^K (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k). \quad (3.15)$$

In this context, the term  $\mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k$  has a particular meaning. It is the difference between the measurement prediction, based on the current estimate of the state, and the actual measurement acquired with the sensor. Thus, it measures how good is the state estimate. Then, by minimizing the sum of all errors between predictions and measurements we seek for the optimal state.

For convenience, let us rewrite the objective function as:

$$F(\mathbf{x}) = \sum_{k=1}^K \underbrace{\mathbf{e}_k(\mathbf{x})^T \boldsymbol{\Omega}_k \mathbf{e}_k(\mathbf{x})}_{e_k(\mathbf{x})}, \quad (3.16)$$

where,

$$\mathbf{e}_k(\mathbf{x}) = \mathbf{h}_k(\mathbf{x}) - \mathbf{z}_k, \quad (3.17)$$

is the error function. If we have an initial estimate  $\check{\mathbf{x}}$  of the state of our system, using Eq. (3.9), we can rewrite a single summand of the objective function as:

$$e_k(\check{\mathbf{x}} + \Delta\mathbf{x}) = (\mathbf{h}_k(\check{\mathbf{x}} + \Delta\mathbf{x}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{h}_k(\check{\mathbf{x}} + \Delta\mathbf{x}) - \mathbf{z}_k) \quad (3.18)$$

$$[\text{plugging in Eq. (3.9)}] \simeq (\mathbf{J}_k \Delta\mathbf{x} + \mathbf{h}_k(\check{\mathbf{x}}) - \mathbf{z}_k)^T \boldsymbol{\Omega}_k (\mathbf{J}_k \Delta\mathbf{x} + \mathbf{h}_k(\check{\mathbf{x}}) - \mathbf{z}_k) \quad (3.19)$$

$$[\text{Eq. (3.17)}] = (\mathbf{J}_k \Delta\mathbf{x} + \mathbf{e}_k)^T \boldsymbol{\Omega}_k (\mathbf{J}_k \Delta\mathbf{x} + \mathbf{e}_k) \quad (3.20)$$

$$[\text{grouping the terms}] = \Delta\mathbf{x}^T \underbrace{\mathbf{J}_k^T \boldsymbol{\Omega}_k \mathbf{J}_k}_{\mathbf{H}_k} \Delta\mathbf{x} - 2 \underbrace{\mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{J}_k}_{\mathbf{b}_k} \Delta\mathbf{x} + \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k \quad (3.21)$$

$$= \Delta\mathbf{x}^T \mathbf{H}_k \Delta\mathbf{x} + 2\mathbf{b}_k \Delta\mathbf{x} + \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k. \quad (3.22)$$

Considering again the summation, we have:

$$F(\check{\mathbf{x}} + \Delta \mathbf{x}) \simeq \sum_{k=1}^K \Delta \mathbf{x}^T \mathbf{H}_k \Delta \mathbf{x} + 2\mathbf{b}_k \Delta \mathbf{x} + \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k \quad (3.23)$$

$$\begin{aligned} \text{[pushing in the summations]} &= \Delta \mathbf{x}^T \underbrace{\left[ \sum_{k=1}^K \mathbf{H}_k \right]}_{\mathbf{H}} \Delta \mathbf{x} + 2 \underbrace{\left[ \sum_{k=1}^K \mathbf{b}_k \right]}_{\mathbf{b}} \Delta \mathbf{x} + \underbrace{\left[ \sum_{k=1}^K \mathbf{e}_k^T \boldsymbol{\Omega}_k \mathbf{e}_k \right]}_c. \end{aligned} \quad (3.24)$$

To wrap up, we rearranged the objective function in Eq. (3.15) to consider its linear approximation around a neighborhood of the initial state  $\check{\mathbf{x}}$ . As we have seen in Eq. (3.11), this depends only on the state increments  $\Delta \mathbf{x}$ . Since our goal is to reach the optimal state  $\mathbf{x}^*$ , we can compute the increments that locally minimize the error and, by repeating this step several times, “hope” to get to the optimum.

The derivative of Eq. (3.24) is:

$$\frac{(\partial \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x} + 2\mathbf{b} \Delta \mathbf{x} + c)}{\partial \Delta \mathbf{x}} = 2\mathbf{H} \Delta \mathbf{x} + 2\mathbf{b}, \quad (3.25)$$

thus, we can find the minimum by solving the linear system:

$$\mathbf{H} \Delta \mathbf{x}^* = -\mathbf{b}, \quad (3.26)$$

and apply it to the current state estimate to get closer to the optimum:

$$\mathbf{x}^* = \check{\mathbf{x}} + \Delta \mathbf{x}^*. \quad (3.27)$$

The whole procedure is the Gauss-Newton (GN) algorithm and is described in detail in Algorithm 1. In general we are not guaranteed to solve the optimization problem in Eq. (3.15). This is because it is ill-posed, since the solution is not unique and its behavior does not change continuously with the initial condition. In practice, this is captured by the event of getting stuck in local minima during the optimization. However, we can have good chances to get to the optimum when we start from a good initial guess and the objective function presents a good convexity.

### 3.1.3 Smooth Manifolds

From the previous section we learned a general technique to approach non-linear optimization problems. As we have seen, starting from an initial estimate of the system state, we build a linear approximation of the error function and then we compute the increments that, applied to the state, locally minimize the error.

It is important to notice that the choice of the state representation is arbitrary. However, it should satisfy at least two requirements: (a) the set of variables should describe in an adequate way the phenomenon under consideration and (b) they should be compatible with the optimization algorithm, that is, the state evolution has to be coherent with the optimization results.

While, a common choice would be to use a set of  $n$  real variables, real world problems often force to consider also non-euclidean topological spaces. As an example,



**Require:**  $\check{\mathbf{x}}$ : initial guess.  $\mathcal{C} = \{\langle \mathbf{z}_k(\cdot), \mathbf{\Omega}_k \rangle\}$ : measurements  
**Ensure:**  $\mathbf{x}^*$ : new solution

- 1: //compute the current error
- 2:  $F_{\text{new}} \leftarrow F$
- 3: // iterate until no substantial improvements are made
- 4: **repeat**
- 5:    $\check{F} \leftarrow F_{\text{new}}$
- 6:    $\mathbf{b} \leftarrow \mathbf{0}$     $\mathbf{H} \leftarrow \mathbf{0}$
- 7:   **for all**  $k = 1 \dots K$  **do**
- 8:     // Compute the prediction  $\hat{\mathbf{z}}_k$ , the error  $\mathbf{e}_k$  and the jacobian  $\mathbf{J}_k$
- 9:      $\hat{\mathbf{z}}_k \leftarrow \mathbf{h}_k(\check{\mathbf{x}})$
- 10:      $\mathbf{e}_k \leftarrow \hat{\mathbf{z}}_k - \mathbf{z}_k$
- 11:      $\mathbf{J}_k \leftarrow \left. \frac{\partial \mathbf{h}_k(\check{\mathbf{x}})}{\partial \Delta \mathbf{x}} \right|_{\mathbf{x}=\check{\mathbf{x}}}$
- 12:     // compute the contribution of this measurement to the linear system
- 13:      $\mathbf{H}_k \leftarrow \mathbf{J}_k^t \mathbf{\Omega}_k \mathbf{J}_k$
- 14:      $\mathbf{b}_k \leftarrow \mathbf{e}_k^t \mathbf{\Omega}_k \mathbf{J}_k$
- 15:     // accumulate the contribution to construct the overall system
- 16:      $\mathbf{H} += \mathbf{H}_k$
- 17:      $\mathbf{b} += \mathbf{b}_k$
- 18:   **end for**
- 19:   // solve the linear system
- 20:    $\Delta \mathbf{x} \leftarrow \text{solve}(\mathbf{H} \Delta \mathbf{x} = -\mathbf{b})$
- 21:   // update the state
- 22:    $\check{\mathbf{x}} += \Delta \mathbf{x}$
- 23:   // compute the new error
- 24:    $F_{\text{new}} \leftarrow F(\check{\mathbf{x}})$
- 25: **until**  $\check{F} - F_{\text{new}} > \epsilon$
- 26: **return**  $\check{\mathbf{x}}$

**Algorithm 1:** Gauss-Newton minimization algorithm

this situation arises when representing the 3D pose of a rigid body: its orientation lies in  $SO(3)$ , the group of rotations about the origin in  $\mathfrak{R}^3$ . A 3D rotation can be described by a  $3 \times 3$  orthonormal matrix.

However, using the 9 matrix parameters as state variables is not feasible since, for the orthonormality constraint, they are not independent. For this reason, a parameterization of minimal dimension is better suited and the typical choice are Euler angles. Unfortunately, treating these parameters as real variables may lead to singularities, *i.e.*, configurations where very large changes in the parameterization are required to represent small changes in the state space.

Since optimization algorithms, like Least-Squares or the Kalman Filter, operate on a local neighborhood of the current estimate, Hertzberg *et al.* [61] proposed to view the space of rotations  $SO(3)$  as a manifold  $S$ . That is, every point of the space has a neighborhood which is homeomorphic to  $\mathfrak{R}^n$ . If we consider the Euler angles as the locally euclidean parameterization, the mapping can be implemented with

two functions:

$$\mathbf{u} = \text{toVector}(\mathbf{R}) \quad (3.28)$$

$$\mathbf{R} = \text{fromVector}(\mathbf{u}), \quad (3.29)$$

where  $\mathbf{u} = [\rho, \theta, \phi]^\top$  is the vector containing the Euler angles and  $\mathbf{R}$  a 3D rotation matrix.

This allows to define compare and modify operations, while retaining the global topology:

$$\boxplus : S \times \mathfrak{R}^n \rightarrow S, \quad [\text{box-plus}] \quad (3.30)$$

$$\boxminus : S \times S \rightarrow \mathfrak{R}^n, \quad [\text{box-minus}] \quad (3.31)$$

where  $\boxplus$  modifies a point in  $S$  by applying a small change expressed in  $\mathfrak{R}^n$ , while  $\boxminus$  expresses in  $\mathfrak{R}^n$  the difference between two points in  $S$ . By applying these operators for the case of 3D rotations we get:

$$\mathbf{u} = \mathbf{R} \boxminus \mathbf{R}_0 = \text{toVector}(\mathbf{R}_0^{-1} \cdot \mathbf{R}), \quad (3.32)$$

$$\mathbf{R} = \mathbf{R}_0 \boxplus \mathbf{u} = \mathbf{R}_0 \cdot \text{fromVector}(\mathbf{u}). \quad (3.33)$$

With this in mind, let us see how to modify the optimization procedure of Sec. 3.1.2 for the case of non-euclidean state spaces. In the remainder, we assume that our state has a redundant parameterization  $\mathbf{X}$  and a minimal one  $\mathbf{x}$ . Likewise, the sensor model can be formulated both as  $\mathbf{h}_k(\mathbf{X})$  and  $\mathbf{h}_k(\mathbf{x})$ .

To deal with non-linear objective functions, we derived a linear approximation of the sensor model in Eq. (3.9). In case of manifold state spaces, this becomes:

$$\mathbf{h}_k(\check{\mathbf{X}} \boxplus \Delta \mathbf{x}) \simeq \mathbf{h}_k(\check{\mathbf{X}}) + \left. \frac{\partial \mathbf{h}_k(\check{\mathbf{X}} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=\mathbf{0}} \cdot \Delta \mathbf{x} \quad (3.34)$$

$$= \mathbf{h}_k(\check{\mathbf{X}}) + \mathbf{J}_k \cdot \Delta \mathbf{x} \quad (3.35)$$

where the linearization point  $\check{\mathbf{X}}$  is fixed. Accordingly, updating the current state estimate after having found the optimal increments  $\Delta \mathbf{x}^*$  results in applying the  $\boxplus$  operator:

$$\check{\mathbf{X}} \leftarrow \check{\mathbf{X}} \boxplus \Delta \mathbf{x}^*. \quad (3.36)$$

If the measurements lie in a non-euclidean space, we need to modify the error function in Eq. (3.17) to take it into account. This time we make use of  $\boxminus$  operator and Eq. (3.17) becomes:

$$\mathbf{e}_k(\mathbf{x}) = \mathbf{h}_k(\mathbf{x}) \boxminus \mathbf{z}_k. \quad (3.37)$$

It is important to remark that this formula is meaningful only when the prediction and the real measurement are “close” enough. Otherwise the assumption of local homeomorphicity to  $\mathfrak{R}^n$  is violated. Finally, if we want to carry on the optimization as in Eq. (3.15), we need to let the error function depend on the state increments:

$$\tilde{\mathbf{e}}_k(\hat{\mathbf{z}}_k \boxplus \Delta \mathbf{z}_k, \mathbf{z}_k) \simeq \tilde{\mathbf{e}}_k(\hat{\mathbf{z}}_k, \mathbf{z}_k) + \underbrace{\left. \frac{\partial \mathbf{e}_k(\hat{\mathbf{z}}_k \boxplus \delta \mathbf{z}_k, \mathbf{z}_k)}{\partial \delta \mathbf{z}_k} \right|_{\delta \mathbf{z}_k=\mathbf{0}}}_{\mathbf{J}_{\mathbf{z}_k}} \Delta \mathbf{z}_k. \quad (3.38)$$

**Require:**  $\check{\mathbf{x}}$ : initial guess.  $\mathcal{C} = \{(\mathbf{z}_k(\cdot), \mathbf{\Omega}_k)\}$ : measurements

**Ensure:**  $\mathbf{x}^*$ : new solution

```

1: //compute the current error
2:  $F_{\text{new}} \leftarrow \check{F}$ 
3: // iterate until no substantial improvements are made
4: repeat
5:    $\check{F} \leftarrow F_{\text{new}}$ 
6:    $\mathbf{b} \leftarrow \mathbf{0}$     $\mathbf{H} \leftarrow \mathbf{0}$ 
7:   for all  $k = 1 \dots K$  do
8:     // Compute the prediction  $\hat{\mathbf{z}}_k$ , the error  $\mathbf{e}_k$  and the jacobians  $\tilde{\mathbf{J}}_k$  and  $\mathbf{J}_{\mathbf{z}_k}$ 
9:      $\tilde{\mathbf{z}}_k \leftarrow \hat{\mathbf{h}}_k(\mathbf{x})$ 
10:     $\tilde{\mathbf{e}}_k \leftarrow \hat{\mathbf{z}}_k \boxminus \mathbf{z}_k$ 
11:     $\mathbf{J}_{\mathbf{z}_k} \leftarrow \left. \frac{\partial \hat{\mathbf{z}}_k \boxminus \mathbf{z}_k}{\partial \delta \mathbf{z}_k} \right|_{\delta \mathbf{z}_k = \mathbf{0}}$ 
12:     $\tilde{\mathbf{J}}_k \leftarrow \left. \frac{\partial \mathbf{e}_k(\mathbf{h}_k(\check{\mathbf{x}} \boxplus \delta \mathbf{x}), \mathbf{z}_k)}{\partial \delta \mathbf{x}_k} \right|_{\delta \mathbf{x}_k = \mathbf{0}}$ 
13:     $\tilde{\mathbf{\Omega}}_k \leftarrow \mathbf{J}_{\mathbf{z}_k} \mathbf{\Omega}_k^{-1} \mathbf{J}_{\mathbf{z}_k}^T$ 
14:    // compute the contribution of this measurement to the linear system
15:     $\mathbf{H}_k \leftarrow \tilde{\mathbf{J}}_k^t \tilde{\mathbf{\Omega}}_k \mathbf{J}_k$ 
16:     $\mathbf{b}_k \leftarrow \mathbf{e}_k^t \tilde{\mathbf{\Omega}}_k \mathbf{J}_k$ 
17:    // accumulate the contribution to construct the overall system
18:     $\mathbf{H} += \mathbf{H}_k$ 
19:     $\mathbf{b} += \mathbf{b}_k$ 
20:  end for
21:  // solve the linear system
22:   $\Delta \mathbf{x} \leftarrow \text{solve}(\mathbf{H} \Delta \mathbf{x} = -\mathbf{b})$ 
23:  // update the state
24:   $\check{\mathbf{x}} \leftarrow \check{\mathbf{x}} \boxplus \Delta \mathbf{x}$ 
25:  // compute the new error
26:   $F_{\text{new}} \leftarrow F(\check{\mathbf{x}})$ 
27: until  $\check{F} - F_{\text{new}} > \epsilon$ 
28: return  $\check{\mathbf{x}}$ 

```

**Algorithm 2:** Gauss-Newton minimization algorithm for manifold measurement and state spaces

Based on this result, we can compute the gaussian approximation of the conditional distribution of the error as

$$p(\mathbf{e}_k | \mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{z}}_k \boxminus \mathbf{z}_k, \underbrace{\mathbf{J}_{\mathbf{z}_k} \mathbf{\Omega}_k^{-1} \mathbf{J}_{\mathbf{z}_k}^T}_{\Sigma_{\mathbf{e}_k | \mathbf{x}}}). \quad (3.39)$$

Note that when we project the measurement to a minimal space through  $\boxminus$ , the conditional covariance of the error  $\Sigma_{\mathbf{e}_k | \mathbf{x}}$  depends on  $\hat{\mathbf{z}}_k = \mathbf{h}_k(\mathbf{x})$ , thus on  $\mathbf{x}$ , and it needs to be recomputed at every iteration.

With these things in place, we can derive a modified version of the Gauss-Newton algorithm, as shown in Algorithm 2.

## 3.2 Local Registration

The last two decades have witnessed important progress in technologies for visual and range data acquisition. Examples are digital monocular cameras, laser range finders and RGB-D cameras. These new devices play the role of enabling technologies for a huge number of engineering applications and have been extensively used in different fields of research like Computer Graphics, Computer Vision and Robotics, just to name a few. In this context, a fundamental problem is geometric registration. It consists in seeking for an alignment between sets of data acquired from different positions and serves as a basis for different processing pipelines.

### 3.2.1 Iterative Closest Point

When an initial guess of the alignment is given or it can be assumed close to the identity, we refer to “local” registration. The Iterative Closest Point (ICP) algorithm is certainly the most adopted method to solve this problem. It refines the estimate of the transform between the input sets by iteratively performing two steps:

1. find corresponding primitives between the two sets
2. compute a transform that minimizes the distance between matching primitives

The procedure is outlined in Algorithm 3, where it is assumed that the matching step has already been executed and the two sets,  $A = \{a_i\}_{i=1,\dots,N}$  and  $B = \{b_i\}_{i=1,\dots,N}$ , are indexed according to their correspondences (*i.e.*,  $a_i$  corresponds with  $b_i$ ).

**Require:** Two sets of data:  $A = \{a_i\}$ ,  $B = \{b_i\}$ ,

Transform initial guess:  $T_0$

**Ensure:** The optimal transform:  $T$

- 1:  $T \leftarrow T_0$ ;
- 2: **while** not converged **do**
- 3:   **for**  $i = 1 : N$  **do**
- 4:      $m_i \leftarrow \text{findClosestInA}(T \cdot b_i)$
- 5:   **end for**
- 6:    $T \leftarrow \underset{T}{\operatorname{argmin}} \sum_i \|T \cdot b_i - m_i\|^2$
- 7: **end while**

**Algorithm 3:** Iterative Closest Point Algorithm

Usually the vanilla algorithm works only in ideal case. Chen and Medioni [18] proposed a variant that’s better suited to real case scenarios. They proposed a different metric to measure point distance, instead of the usual euclidean norm between the points they adopted a *point-to-plane* metric. This consists in projecting the point distance along the surface normal, so that, line 6 of Algorithm 3 becomes:

$$T \leftarrow \underset{T}{\operatorname{argmin}} \sum_i \|\eta_i \cdot (T \cdot b_i - m_i)\|^2, \quad (3.40)$$

where  $\eta_i$  is the surface normal at  $m_i$ .

Segal *et al.* [135] further extended this approach by proposing a general distance metric, based on a probabilistic viewpoint, that captures several ICP variants, like: *point-to-point*, *point-to-plane* and *plane-to-plane*. To derive their formulation, they start from the consideration that the two data sets can be thought of as sampled from the underlying real data,  $\hat{A} = \{\hat{a}_i\}$  and  $\hat{B} = \{\hat{b}_i\}$ , according to a normal distribution, *i.e.*,  $a_i \sim \mathcal{N}(\hat{a}_i, C_i^A)$  and  $b_i \sim \mathcal{N}(\hat{b}_i, C_i^B)$ . Therefore, if we set  $d_i(T) = b_i - T \cdot a_i$ , line 6 of Algorithm 3 becomes:

$$T \leftarrow \operatorname{argmin}_T \sum_i -\log(p(d_i(T))). \quad (3.41)$$

Since  $a_i$  and  $b_i$  are normally distributed,  $d_i(T)$  is still a gaussian and the previous equation can be rewritten as:

$$T \leftarrow \operatorname{argmin}_T \sum_i d_i(T)^\top (C_i^B + TC_i^A T^\top)^{-1} d_i(T). \quad (3.42)$$

From this formula it is easy to verify that when:

$$C_i^B = \mathbf{I}_{3 \times 3} \quad (3.43)$$

$$C_i^A = \mathbf{0}_{3 \times 3} \quad (3.44)$$

we obtain the *point-to-point* metric. If:

$$C_i^B = \mathbf{P}_i^{-1} \quad (3.45)$$

$$C_i^A = \mathbf{0}_{3 \times 3} \quad (3.46)$$

with  $\mathbf{P}_i$  orthogonal projection matrix, we get the *point-to-plane* metric. Finally, with:

$$C_i^B = \mathbf{R}_{\mu_i} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{\mu_i}^\top \quad (3.47)$$

$$C_i^A = \mathbf{R}_{\nu_i} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{\nu_i}^\top \quad (3.48)$$

Eq. (3.42) implements the *plane-to-plane* metric, where  $\mathbf{R}_{\mu_i}$  and  $\mathbf{R}_{\nu_i}$  are the 3D rotation matrices built from the surface normals.

### 3.2.2 Least-Squares Solution

So far, we discussed distance metrics for the ICP algorithm but nothing has been said on how to actually minimize that distance. With the tools presented in Sec. 3.1, we can approach local registration as a state estimation problem.

#### State

If we consider the general case, where we want to align surfaces, our state space will be the Lie Group  $SE(3)$  of 3D rigid transformations. Since this topological group is

a smooth manifold, according to Sec. 3.1.3, we provide both parameterizations of the state:

$$\mathbf{X} \in SE(3) : [\mathbf{R} \mid \mathbf{t}] \quad [\text{full}] \quad (3.49)$$

$$\mathbf{x} \in \mathfrak{R}^6 : \underbrace{(x \ y \ z)}_{\mathbf{t}} \underbrace{(\alpha_x \ \alpha_y \ \alpha_z)}_{\mathbf{u}}^\top \quad [\text{minimal}] \quad (3.50)$$

with:

- $\mathbf{R} \in SO(3)$ , 3D rotation matrix
- $\mathbf{t} \in \mathfrak{R}^3$ , translation vector
- $\mathbf{u} \in \mathfrak{R}^3$ , Euler angles.

The mapping between these two representations is implemented as in Eq. (3.28) and Eq. (3.29). Thus, a perturbation to the state is performed as:

$$\mathbf{X} \boxplus \Delta \mathbf{x} = \text{fromVector}(\Delta \mathbf{x}) \mathbf{X} \quad (3.51)$$

$$= [\mathbf{R}(\Delta \mathbf{u}) \mathbf{R} \mid \mathbf{R}(\Delta \mathbf{u}) \mathbf{t} + \Delta \mathbf{t}]. \quad (3.52)$$

### Error Function

In the standard ICP algorithm the sets to be aligned are point clouds. Let us assume the usual convention and call them *model* and *data*. In a robotic mapping scenario, the *model* point cloud is the map that the robot has built up to a certain time instant, while the *data* point cloud is a new measurement acquired with its sensors at that time instant. It follows that an effective and sound parameterization for the measurements is a 3D vector, that is:

$$\mathbf{z}_k \in \mathfrak{R}^3. \quad (3.53)$$

Consequently, we can define an *observation model*. This is a function that predicts a virtual measurement, given the current state estimate. For an euclidean measurement, the prediction function is:

$$\mathbf{h}_k(\mathbf{X}) = \mathbf{R} \cdot \mathbf{p}_k + \mathbf{t}, \quad (3.54)$$

where  $\mathbf{p}_k$  is the point in the *model* that matches with  $\mathbf{z}_k$  in the *data*. Therefore, the error function is:

$$\mathbf{e}_k(\mathbf{X} \boxplus \Delta \mathbf{x}) = \mathbf{h}_k(\mathbf{X} \boxplus \Delta \mathbf{x}) - \mathbf{z}_k \quad (3.55)$$

$$= \mathbf{R}(\Delta \mathbf{u}) \underbrace{[\mathbf{R} \cdot \mathbf{p}_k + \mathbf{t}]}_{\mathbf{p}'_k} + \Delta \mathbf{t} - \mathbf{z}_k, \quad (3.56)$$

and the Jacobian:

$$\mathbf{J}_k = \left. \frac{\partial \mathbf{e}_k(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0} \quad (3.57)$$

$$= \left( \frac{\partial \mathbf{e}_k(\cdot)}{\partial \Delta \mathbf{t}} \quad \frac{\partial \mathbf{e}_k(\cdot)}{\partial \Delta \mathbf{u}} \right) \Big|_{\Delta \mathbf{x}=0} \quad (3.58)$$

$$= \left( \frac{\partial \Delta \mathbf{t}}{\partial \Delta \mathbf{t}} \quad \frac{\partial \mathbf{R}(\Delta \mathbf{u}) \mathbf{p}'_k}{\partial \Delta \mathbf{u}} \right) \Big|_{\Delta \mathbf{x}=0} \quad (3.59)$$

$$= (\mathbf{I} \ [\mathbf{p}'_k]_\times). \quad (3.60)$$

### 3.3 Global Optimization

In the previous section we have seen how the powerful Bayesian State Estimation framework can be adopted to cast the local registration problem into a minimization one and how to solve it with the Least-Squares method. For its generality, the same framework is also used to solve another central problem in Robotics, namely, Simultaneous Localization and Mapping and we will see that, despite some differences with the registration setting, the same methodology provides an effective tool to tackle this problem.

In the remainder, we will focus on the smoothing approach for SLAM and present a technique, based on factor graphs, that provides an efficient solution.

#### 3.3.1 Multi-Point Registration

Similar to Sec. 3.1.1, our goal is to estimate the probability distribution over the possible states of our system given a set of measurements (see Eq. (3.2)). For the Multi-Point Registration problem we consider the following setting. A robot explores an unknown environment populated with distinguishable landmarks and observes them with a 3D sensor. For the robot poses  $\mathbf{X}_{1:N}^r$  we choose the usual space of 3D rigid transformations, *i.e.*,  $\mathbf{X}_i^r \in SE(3)$ . The landmarks  $\mathbf{x}_{1:M}^l$  are 3D points, with  $\mathbf{x}_j^l \in \mathfrak{R}^3$ . At each robot pose the sensor returns a set of observations, where an observation  $\mathbf{z}_{i,j} \in \mathfrak{R}^3$  consists in the perceived landmark  $\mathbf{x}_j^l$  expressed in the robot local frame  $\mathbf{X}_i^r$ . We want to estimate the robot poses  $\mathbf{x}_{1:n}^r$  and the landmark positions  $\mathbf{x}_{1:m}^l$ , given all the measurements  $\{\mathbf{z}_{i,j}\}$ . Again, we follow the methodology presented in Sec. 3.1 to carry on the optimization.

#### State

The state  $\mathbf{X}$  is composed by the robot poses and the landmark positions:

$$\mathbf{X} : \mathbf{X} = \{\mathbf{X}_r^{[1]}, \dots, \mathbf{X}_r^{[M]}, \mathbf{x}_i^{[1]}, \dots, \mathbf{x}_i^{[M]}\} \quad (3.61)$$

$$\mathbf{X}_r^{[n]} \in SE(3) : \mathbf{X}_r^{[n]} = \left( \mathbf{R}^{[n]} | \mathbf{t}^{[n]} \right) \quad (3.62)$$

$$\mathbf{x}_i^{[m]} \in \mathfrak{R}^3 : \mathbf{x}_i^{[m]} = \left( x^{[m]} \ y^{[m]} \ z^{[m]} \right)^\top \quad (3.63)$$

where we inverted subscript and superscript for readability. It follows that the perturbation of the state will be a vector containing the proper minimal perturbation for each state variable:

$$\Delta \mathbf{x} \in \mathfrak{R}^{6N+3M} : \Delta \mathbf{x} = \left( \Delta \mathbf{x}_r^{[1]\top}, \dots, \Delta \mathbf{x}_r^{[N]\top}, \Delta \mathbf{x}_i^{[1]\top}, \dots, \Delta \mathbf{x}_i^{[M]\top} \right)^\top \quad (3.64)$$

$$\Delta \mathbf{x}_r^{[n]\top} \in \mathfrak{R}^6 : \Delta \mathbf{x}_r^{[n]\top} = \underbrace{\left( \Delta x^{[n]} \ \Delta y^{[n]} \ \Delta z^{[n]} \right)}_{\Delta \mathbf{t}^{[n]}} \underbrace{\left( \Delta \alpha_x^{[n]} \ \Delta \alpha_y^{[n]} \ \Delta \alpha_z^{[n]} \right)^\top}_{\Delta \alpha^{[n]}} \quad (3.65)$$

$$\Delta \mathbf{x}_i^{[m]\top} \in \mathfrak{R}^3 : \Delta \mathbf{x}_i^{[m]\top} = \underbrace{\left( \Delta x^{[m]} \ \Delta y^{[m]} \ \Delta z^{[m]} \right)^\top}_{\Delta \mathbf{t}^{[m]}}. \quad (3.66)$$

As in Eq. (3.52), the perturbation is applied with the  $\boxplus$  operator:

$$\mathbf{X}' = \mathbf{X} \boxplus \Delta \mathbf{x} \quad (3.67)$$

$$\mathbf{X}_r^{[n]'} = \Delta \mathbf{x}_r^{[n]} \boxplus \mathbf{X}_r^{[n]} \quad (3.68)$$

$$= v2t(\Delta \mathbf{x}_r^{[n]}) \mathbf{X}_r^{[n]} \quad (3.69)$$

$$\mathbf{X}_l^{[m]'} = \mathbf{X}_l^{[m]} + \Delta \mathbf{x}_l^{[m]}, \quad (3.70)$$

by taking care of incrementing each state block accordingly.

### Error Function

The measurement of landmark  $m$  from robot pose  $n$  is:

$$\mathbf{z}^{[n,m]} \in \mathfrak{R}^3 : \mathbf{z}^{[n,m]} = \left( x^{[n,m]} \ y^{[n,m]} \ y^{[n,m]} \right)^\top \quad (3.71)$$

In this case, the prediction and error functions resemble those of the ICP algorithm:

$$\mathbf{h}^{[n,m]}(\mathbf{X}) = \mathbf{X}_r^{[n]} \mathbf{x}_l^{[m]} \quad (3.72)$$

$$\mathbf{e}^{[n,m]}(\mathbf{X}) = \mathbf{X}_r^{[n]} \mathbf{x}_l^{[m]} - \mathbf{z}^{[n,m]} \quad (3.73)$$

$$\mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x}) = v2t(\Delta \mathbf{x}_r^{[n]}) \mathbf{X}_r^{[n]} (\mathbf{X}_l^{[m]} + \Delta \mathbf{x}_l^{[m]}) - \mathbf{z}^{[n,m]} \quad (3.74)$$

The difference is that  $\mathbf{h}^{[n,m]}(\mathbf{X})$  depends only on robot pose  $n$  and landmark  $m$ . The consequence is that the Jacobian  $\mathbf{J}^{[n,m]}$  will be non-zero only in correspondence of these block indices:

$$\frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} = \left( \dots \mathbf{0}_{3 \times 6} \dots \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_r^n} \dots \mathbf{0}_{3 \times 6} \right. \\ \left. \mathbf{0}_{3 \times 3} \dots \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_l^m} \dots \mathbf{0}_{3 \times 3} \dots \right) \quad (3.75)$$

$$\hat{\mathbf{z}}^{[n,m]} = \mathbf{X}_r^{[n]} \mathbf{x}_l^{[m]} \quad (3.76)$$

$$\left. \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_r^n} \right|_{\Delta \mathbf{x}_r^{[n]} = \mathbf{0}} = \underbrace{\left( \mathbf{I}_{3 \times 3} \mid [-\hat{\mathbf{z}}^{[n,m]}]_\times \right)}_{\mathbf{J}_r^{n,m}} \quad (3.77)$$

$$\left. \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_l^m} \right|_{\Delta \mathbf{x}_l^{[m]} = \mathbf{0}} = \underbrace{\mathbf{R}^{[n]}}_{\mathbf{J}_l^{n,m}} \quad (3.78)$$

$$\mathbf{J}^{[n,m]} = \left( \dots \mathbf{0}_{3 \times 6} \dots \mathbf{J}_r^{[n,m]} \dots \mathbf{0}_{3 \times 6} \mathbf{0}_{3 \times 3} \dots \mathbf{J}_l^{[n,m]} \dots \mathbf{0}_{3 \times 3} \dots \right) \quad (3.79)$$

### 3.3.2 Factor Graphs and Sparse Least-Squares

As promised, we again used the Bayesian framework to address an estimation problem, namely, Multi-Point Registration. Therefore, the final goal is to estimate a posterior probability of this form:

$$p(\mathbf{x} \mid \mathbf{z}) = p(\mathbf{x}_{1:N} \mid \mathbf{z}_{1:M}). \quad (3.80)$$



For a real-world problem, we immediately notice that, having defined the state as in Eq. (3.61), we have to deal with a complex task. For this reason, we seek for mathematical manipulations and simplifying assumptions that allow to turn the said problem in a more tractable form. In this context, we will see that by leveraging on concepts borrowed from Graph Theory it is possible to represent the problem in a way that allows to have a better insight and, as a consequence, to solve it more efficiently.

The Multi-Point Registration is a Maximum A Posteriori (MAP) estimation problem. That is, we want to find the state  $\mathbf{x}^*$  that maximizes Eq. (3.80):

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{x} | \mathbf{z}) \quad (3.81)$$

$$\text{[Bayes rule]} = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{z} | \mathbf{x})p(\mathbf{x}) \quad (3.82)$$

$$\text{[Independence]} = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{x}) \prod_k p(\mathbf{z}_k | \mathbf{x}_k) \quad (3.83)$$

$$= \underset{\mathbf{x}}{\operatorname{argmin}} - \log \underbrace{\left( p(\mathbf{x}) \prod_k p(\mathbf{z}_k | \mathbf{x}_k) \right)}_{\mathbf{F}(\mathbf{x})} \quad (3.84)$$

$$= \underset{\mathbf{x}}{\operatorname{argmin}} \mathbf{F}(\mathbf{x}). \quad (3.85)$$

Where, under the gaussian assumption as in Eq. (3.13), we can rewrite the objective function as:

$$\mathbf{F}(\mathbf{x}) = \sum_k \mathbf{e}_k(\mathbf{x}_k, \mathbf{z}_k)^\top \boldsymbol{\Omega}_k \mathbf{e}_k(\mathbf{x}_k, \mathbf{z}_k). \quad (3.86)$$

Now, let us delve deeper in Eq. (3.86). The state  $\mathbf{x} = (\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top)^\top$  is a vector of parameters, each  $\mathbf{x}_i$  can be seen as a parameter block. The subscript  $k$  refers to the  $k^{\text{th}}$  constraint in the objective function that involves a subset of the state  $\mathbf{x}_k = (\mathbf{x}_{k_1}^\top, \dots, \mathbf{x}_{k_q}^\top)^\top \subset \mathbf{x} = (\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top)^\top$  and has mean  $\mathbf{z}_k$  and information matrix  $\boldsymbol{\Omega}_k$ . As usual, the error function  $\mathbf{e}_k(\mathbf{x}_k, \mathbf{z}_k) = \mathbf{h}_k(\mathbf{x}_k) - \mathbf{z}_k$  measures how close is the prediction from parameter blocks in  $\mathbf{x}_k$  to the actual measurement  $\mathbf{z}_k$ .

From our knowledge of the problem, we recall that in a single constraint only a small subset of the parameters vector is interested, *e.g.* a single landmark measurement  $\mathbf{z}^{[n,m]}$  is affected only by the robot pose  $\mathbf{X}_r^{[n]}$  and the landmark position  $\mathbf{x}_i^{[m]}$ . This allows a graphical representation of the posterior distribution that highlights the correlation between the problem variables. The measurements independence assumption (Eq. (3.83)) implies a factorization of the PDF. Thus, we can use a factor graph to represent it. The nodes of this graph are the state parameters, while an edge among the nodes represents a constraint between the linked nodes. This representation allows to have a visual feedback on the structure of the problem.

From Eq. (3.21), we know that the matrix  $\mathbf{H}$  and the vector  $\mathbf{b}$  are built by summing the contributions deriving from each constraint:

$$\mathbf{H} = \sum_{k \in \mathcal{C}} \mathbf{H}_k. \quad (3.87)$$

$$\mathbf{b} = \sum_{k \in \mathcal{C}} \mathbf{b}_k \quad (3.88)$$



**Part II**

**Metric maps**



## Chapter 4

# Taxonomy of Metric Map Representations

The main purpose of SLAM algorithms is to recover the geometric structure of an environment. Such representation is referred to as “metric map”. Informally speaking, the term *metric* means that it is a 1 : 1 scale representation of the environment and, thus, can be used to measure distance to goals or avoid real obstacles.

A metric map is built with an incremental process. This consists in integrating into a global reference frame the sequence of sensor measurements acquired by the robot during navigation. As depicted in Fig. 4.1, this process can be seen as the continuous iteration of two steps: registration and fusion.

When a robot acquires a new measurement, its percepts (points for range sensors and intensities for visual sensors) will be expressed in the local reference frame attached to the sensor. Registration is used to find an alignment between the new data and the map built so far, to transform it from the local to the global reference frame. After that, it is possible to update the global map.

Simply adding these incoming percepts to the map is rarely a good idea, since it results in a lot of redundant information and a non-negligible consumption of memory. However, consecutive views of the environment must present a certain overlap to be registered. With a *data association* procedure it is possible to find pair of percepts among consecutive views generated from the same real-world element. Therefore, associated ones are fused together, usually taking some form of average, while un-associated ones are added to the map, for being considered as new information.

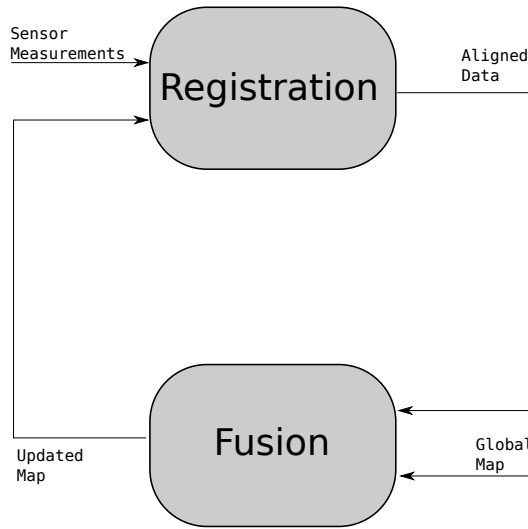
To represent an object surface there are different possible ways. From Differential Geometry we learn that a surface  $\mathbf{S}$  can be expressed in three analytical forms [33]:

**explicit** - when it is described by the set of primitives  $\mathbf{p}_i$  that compose it:

$$\mathbf{S} = \{\mathbf{p}_i\} \quad (4.1)$$

**parametric** - in this case it is identified as a mapping between a domain  $\Omega \in \mathfrak{R}^2$  and points in  $\mathfrak{R}^3$ :

$$\mathbf{S} : \Omega \in \mathfrak{R}^2 \rightarrow \mathfrak{R}^3 \quad (4.2)$$



**Figure 4.1.** Incremental map building process.

**implicit** - if it is defined on a scalar field as the set of points that satisfy a certain condition:

$$\mathbf{S} = \{(x, y, z) \mid f(x, y, z) = 0\} \quad (4.3)$$

In robotics applications, it is very unlikely to recover an analytical form of the scene geometry. Nevertheless, the digital representations built in robotics can be considered as good approximations. Indeed, from the previous definitions we may notice that: point clouds and voxel grids can be considered as explicit representations, digital elevation maps (dem) as parametric and signed distance functions as implicit.

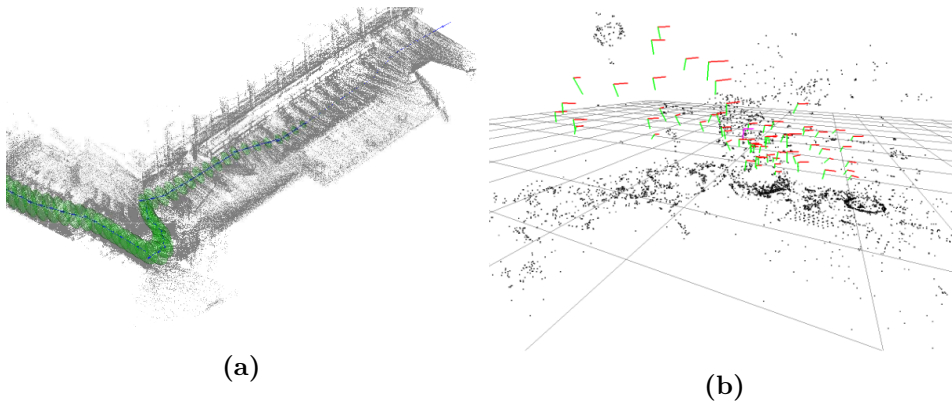
Apart from dems which, in general, are not suited for 3D geometry, in the remainder we will provide a categorization of the various representations proposed in literature along with a final comparison.

## 4.1 Sparse Representation

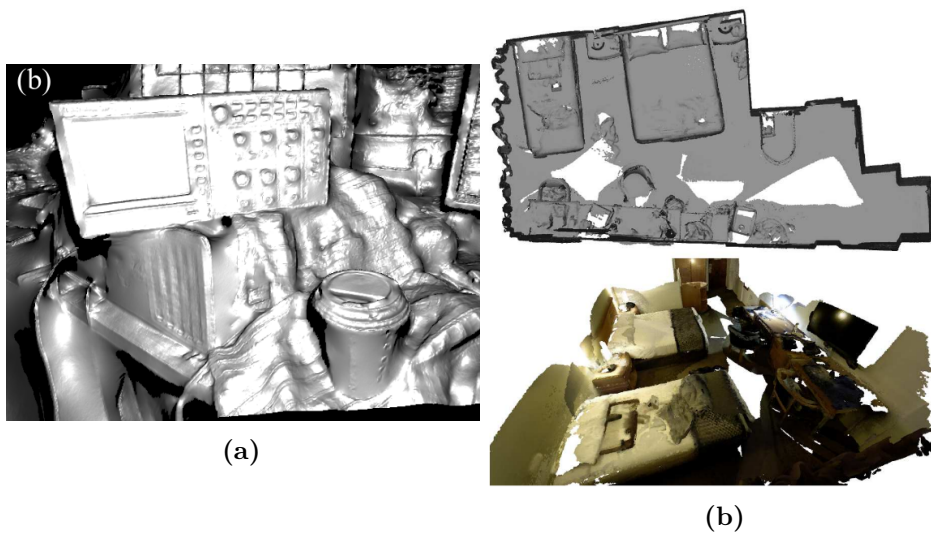
As we have seen in Sec. 2.1.1, it is possible to align two surfaces by considering only a subset of interesting points from the two sets of data. This approach is typically used for data coming from monocular or stereo cameras [100, 78] and in Computer Vision it is usually referred as Structure from Motion [58]. In this way, the scene is represented with a set of *sparse* 3D landmarks that usually correspond to distinguishable elements of the environment.

Sparse point clouds obtained success in robotics applications because they can be obtained from the output of range sensors [22] and, as we have seen in Sec. 2.1.2, common techniques exist to align this data.

Another reason why this representation has been adopted by the majority of mapping systems, is because with a simple SLAM formulation it is still possible to achieve robust sensor tracking and loop detection. Furthermore, as will be shown in Chapters 5 and 6, there is an ongoing line of research to extend these methods for including other geometric primitives as landmarks, *i.e.*, lines and planes.



**Figure 4.2.** Example of sparse representations: (a) image from [22] (b) image from [76].

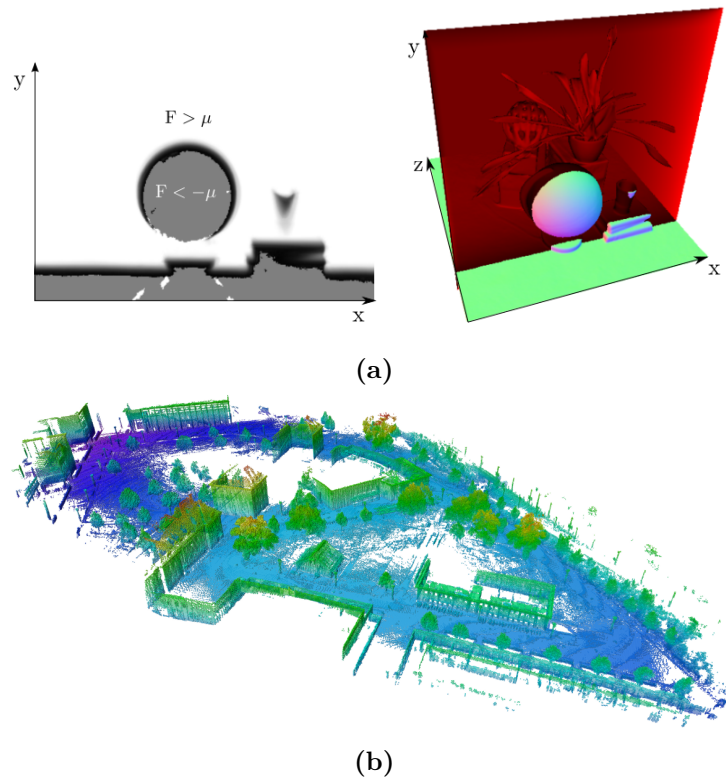


**Figure 4.3.** Example of dense representations: (a) image from [102] (b) image from [170].

## 4.2 Dense Representation

Despite their efficiency, *landmark-based* methods present two main limitations: (1) they heavily depend on the feature extraction process and (2) they provide a rather poor representation of the environment since they tend to filter out featureless portions.

To provide detailed 3D models of the environment that are better suited for visualization and rendering, methods based on *dense* representations have been proposed. They work by considering the whole output of 3D sensors, like Laser Scanners or RGB-D cameras, for reconstruction. As a consequence, they represent the scene with large unstructured sets of points [102] or surfels [137]. Of course, this comes at the price of an higher computational cost.



**Figure 4.4.** Example of volumetric representations: (a) image from [103] (b) image from [64].

### 4.3 Volumetric Representation

Another line of investigation is to adopt structured representations, such as the ones used in Computer Graphics for Surface Reconstruction [6]. These are called "boundary representations", since they define 3D objects in terms of their surface boundary.

In this context, implicit surface representation has gained increasing attention. This is caused by two main reasons: first, they can represent a surface of arbitrary genus and second, their data structure is naturally suited for geometry processing like Boolean operations and Dynamics simulation. A common implicit representation of surfaces is the *Signed Distance Function* (SDF), that can be estimated with different methods [26, 63].

Other relevant volumetric representations are based on spatial-partitioning. The most common technique is to decompose the 3D space into identical cubes (voxels), arranged in a regular 3D grid. Since this representation scales poorly, adaptive ones have been proposed, like: hierarchical data-structures (octrees [174] or  $N^3$  trees [17]) and flat hash-tables [106].



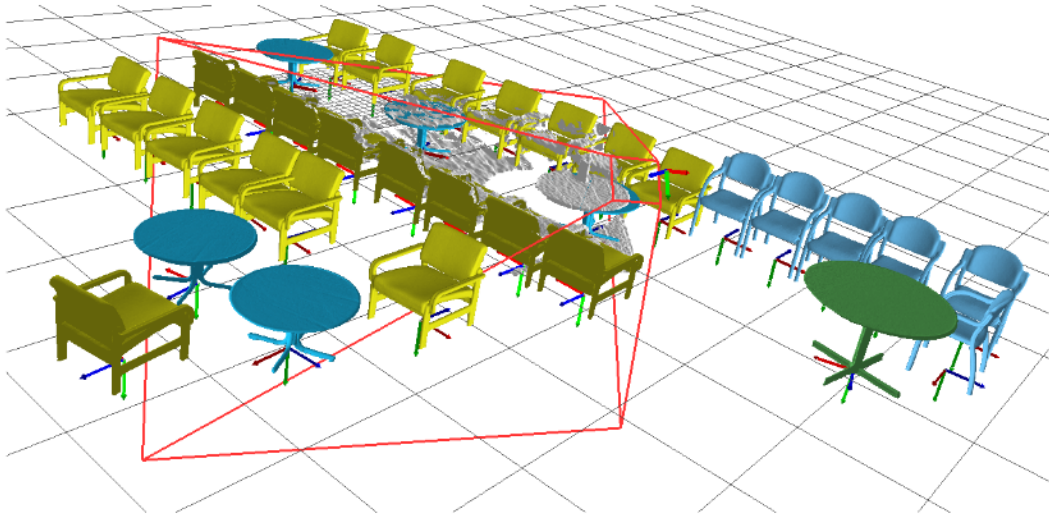


Figure 4.5. Example of object-based representation, image from [132].

## 4.4 Object-based Representation

Despite all the representations mentioned so far can be considered as an approximation to the scene geometry, they are all built from low-level primitives. This limits the robot planning and reasoning capabilities, since we human perceive the environments in terms of the objects it is made of.

The solution proposed in the seminal works by Moreno *et al.* [132], Civera *et al.* [21] and Dame *et al.* [28] is to investigate *object-based* representations, where the map is made of objects and solid shapes as it is the case in Computer Aided Design (CAD).

Of course, building such a map is an expensive procedure and requires many simplifying assumptions. Nevertheless, still this type of representation is very attractive because it naturally lends itself to represent also objects semantic and physical properties, which can be essential for the robot to execute its tasks.

## 4.5 Comparison

To wrap up, in Tab. 4.1 we propose a comparison of the presented approaches based on four indices:

**Robustness:** it is a measure of the pose tracking capabilities. Sparse methods can be considered robust because they rely on distinguishable features of the environment and can count on efficient loop detection. On the other side, dense and SDF-based methods achieve robustness by exploiting the rich source of information coming from range sensors.

**Accuracy:** it describes how well the reconstructed model approximates the real surface. Occupancy-based representations can be used for navigation but are too coarse to describe objects shapes. While other representations, apart from sparse ones, are better indicated to recover the scene geometry.

**Complexity:** it refers to the processing cost, in terms of computational complexity. Sparse methods can be considered the most efficient ones because they work on a compact representation and are the most scalable. Other methods need to process a large amount of data (dense) and/or perform complex pre and post-processing (volumetric, object-based).

**Usability:** it indicates the application possibilities of the reconstructed map. Dense and SDF representations provide detailed models but can be used only for visualization. While occupancy and object-based representations are suitable for task planning and execution.

		Robustness	Accuracy	Complexity	Usability
Sparse		✓✓	✓	✓✓	✗✗
Dense		✓✓	✓✓	✗✗	✓
Volumetric	Occupancy	✓	✗	✗	✓✓
	SDF	✓✓	✓✓	✗✗	✓
Object-based		✓	✓✓	✗✗	✓✓

**Table 4.1.** Comparison of metric representations.

## Chapter 5

# Unifying Local Registration Algorithms

As we have seen in Chapter 2, the spread of *landmark-based* SLAM approaches is mainly due to the existence of well-known techniques for detecting, matching and registering point primitives. Despite their efficiency, these techniques have known drawbacks: detection may fail in textureless scenes, matching is hindered by the features low descriptiveness and registration may be unable to recover large rotations. In all the above mentioned situations, pose tracking is likely to yield poor results.

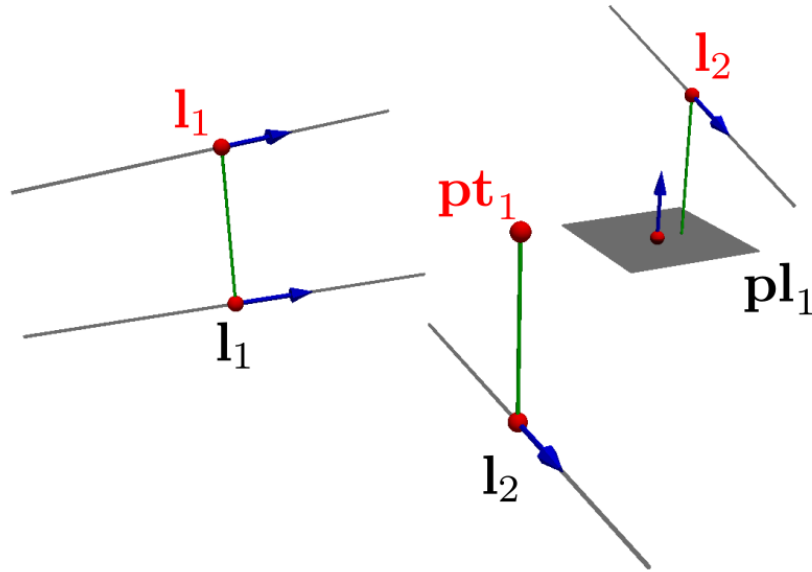
At the same time, man made environment are rich in structure. Planar items such as walls, floors and tables cover a great portion of the environment. Similarly, the edges of these planar structures often form lines. In this chapter we propose a unified representation for different types of primitives such as points, lines and planes. In this way, it is possible to devise a registration algorithm for aligning hybrid scenes in a single formulation. Moreover, we can define correspondences among items in the scenes that belong to different classes: besides enforcing that two planes or two points in the scenes are the same, we can also express constraints such as “a point lies on a line”, or “a line lies on a plane”.

Exploiting structure to describe a scene has clear advantages on the storage: describing a room as a set of walls and a floor requires far less memory than storing the corresponding point cloud. In turn, registration algorithms might gain in efficiency and display larger alignment capability coming from the higher descriptiveness of the primitives.

The derivations proposed in this chapter capture in a uniform manner several Iterative Closest Point (ICP) variants [7, 135, 138]. Our system benefits from the structure when present, while it degrades to regular ICP in absence of structure. Furthermore, it provides compact models that can be used to reproduce the geometry of the scene.

### 5.1 Generalized Local Registration

Registration aims at finding the transform that better aligns two scenes through non-linear optimization of an objective function. As a consequence, the choice of this function will have a strong impact on the final solution. In general, convex objective






**Figure 5.1.** A typical scenario addressable with the proposed representation: we want to register a moving scene (red) onto a fixed scene (black). The scenes are composed by points ( $\mathbf{pt}$ ), lines ( $\mathbf{l}$ ) and planes ( $\mathbf{pl}$ ). Green lines indicate the constraints between two geometric entities. The proposed representation allows to model both homogeneous (e.g. line-line) and heterogeneous (e.g. point-plane) constraints.

functions are preferred, since they lead to an easier convergence. In practice, the objective functions used for registration problems are not convex, thus it is likely to fall in local minima.

With the proposed formulation, we can model not only point-point correspondences but any of the nine possible pairings between points, lines and planes (see Fig 5.1). This allows to exploit the higher descriptiveness of geometric primitives such as lines and planes for widening the objective function convergence basin and, consequently, avoiding local minima.

Of course, a straightforward approach is to define an *ad-hoc* error function for each type of correspondence (e.g., point-point, point-line, line-plane and so on). Then, for each of them, a different optimization has to be formulated. Our approach is to define a single parametrization for different geometric primitives. Therefore, once a primitive is represented as a unified entity, all the other computation modules are agnostic of the primitive's type.

In the remainder of this section, we first introduce our representation of primitives. Subsequently, we define how to apply an isometry to a scene. With this basic operation defined, we show how to define an error function between two primitives that are connected by a constraint. Finally, we propose both an iterative and a direct implementation of a solver capable of finding the optimal transformation between two scenes, given a set of correspondences.

type	$\mathbf{p}_m$	$\mathbf{R}_m$	$\mathbf{\Omega}_m$	Shape of $\mathbf{\Omega}_m$
point	$\mathbf{p}$	$\mathbf{R}$	$\text{diag}(1, 1, 1)$	
line	$\mathbf{p}_l$	$\mathbf{R}_l$	$\text{diag}(0, 1, 1)$	
plane	$\mathbf{p}_\pi$	$\mathbf{R}_\pi$	$\text{diag}(1, 0, 0)$	

**Table 5.1.** Matchables table. The shape of  $\mathbf{\Omega}_m$  discriminates the type of primitive represented by the matchable. The confidence ellipsoid obtained from  $\mathbf{\Omega}_m$  is a sphere if the matchable is a point. If the primitive is a line or a plane the confidence ellipsoid degenerates respectively to a cylinder or to two parallel planes.

### 5.1.1 Representation

Our representation stems from the fact that points, lines and planes can be defined in terms of *degenerate quadrics*. In general, a quadric can be described by the following equation:

$$(\mathbf{x} - \mathbf{p})^\top \mathbf{A} (\mathbf{x} - \mathbf{p}) = 0. \quad (5.1)$$

Here,  $\mathbf{p} \in \mathbb{R}^3$  is the origin of the quadric and  $\mathbf{A}$  a symmetric matrix. A point  $\mathbf{x} \in \mathbb{R}^3$  lies on the quadric if it satisfies Eq. (5.1).  $\mathbf{A}$  can be factorized as  $\mathbf{A} = \mathbf{R}\mathbf{\Omega}\mathbf{R}^\top$ , where the columns of  $\mathbf{R}$  are the axes of the quadric, and  $\mathbf{\Omega}$  is a diagonal matrix containing the eigenvalues of  $\mathbf{A}$ . The shape of the quadric is entirely determined by its eigenvalues. In the remainder, we refer to the first column of  $\mathbf{R}$  as the *direction*  $\mathbf{d}$  of a primitive.

Points can be represented as spheres with a null radius, thus all eigenvalues should be equal and positive:  $\mathbf{\Omega} = \text{diag}(1, 1, 1)$ . Lines can be represented as cylinders having null radius, thus, assuming the cylinder axis is parallel to the direction  $\mathbf{d}$ , the corresponding omega is  $\mathbf{\Omega} = \text{diag}(0, 1, 1)$ . Finally, planes can be represented through quadric as two matching planes. In this case, being the plane normal parallel to  $\mathbf{d}$ ,  $\mathbf{\Omega} = \text{diag}(1, 0, 0)$ .

The reader might notice that the  $\mathbf{\Omega}$  used to represent the above primitives can be scaled arbitrarily by any positive number. Setting the non-null eigenvalues to 1 has however the effect of turning the value of the left hand side  $\|\mathbf{x} - \mathbf{p}\|_{\mathbf{A}}^2$  of Eq. (5.1) to the squared euclidean distance between  $\mathbf{x}$  and the closest point on the quadric. Tab. 5.1 summarizes the parameters used for each type of primitive.

In our system we call such quadrics *matchables*. For a matchable  $\mathbf{m}$  we store: the origin  $\mathbf{p}_m$ , the rotation matrix  $\mathbf{R}_m$  and the eigenvalues matrix  $\mathbf{\Omega}_m$ :

$$\mathbf{m} : \langle \mathbf{p}_m, \mathbf{R}_m, \mathbf{\Omega}_m \rangle. \quad (5.2)$$

### 5.1.2 Transformation

Let  $\mathbf{X} = [\mathbf{R}_x | \mathbf{t}_x] \in SE(3)$  be a transformation. The operation of applying a transformation to a matchable  $\mathbf{m}$  results in a new matchable  $\mathbf{m}' = \mathbf{X} \cdot \mathbf{m}$  with the

following parameters:

$$\mathbf{m}' = \begin{pmatrix} \mathbf{R}_x \mathbf{p}_m + \mathbf{t}_x \\ \mathbf{R}_x \mathbf{R}_m \\ \Omega_m \end{pmatrix}. \quad (5.3)$$

Here we parametrized  $\mathbf{X}$  as a rotation matrix  $\mathbf{R}_x \in SO(3)$  and a translation vector  $\mathbf{t}_x \in \mathbb{R}^3$ . Intuitively, applying an isometry to a matchable results in a transformation of the Euclidean component  $\mathbf{p}_m$  and in a rotation of the matrix  $\mathbf{R}_m$ , leaving unchanged  $\Omega_m$ .

### 5.1.3 Distance

In this section we present a metric to evaluate a distance  $e(\mathbf{m}, \mathbf{m}')$  between a pair of matchables  $\mathbf{m}$  and  $\mathbf{m}'$ . If  $\mathbf{m}$  is a point and  $\mathbf{m}'$  is any primitive Eq. (5.1) allows us to compute the squared distance between a point and the quadric in  $\mathbf{m}'$  as:

$$e(\mathbf{m}, \mathbf{m}') = (\mathbf{p}_m - \mathbf{p}_{m'})^\top \mathbf{A}_{m'} (\mathbf{p}_m - \mathbf{p}_{m'}) \quad (5.4)$$

To compute the distance between two planes (or, equivalently, two lines) we need to consider in the difference their directions  $\mathbf{d}_m$  and  $\mathbf{d}_{m'}$ . Adding to Eq. (5.4) a quadratic term  $\|\mathbf{d}_m - \mathbf{d}_{m'}\|^2$  that captures the difference in directions results in a metric that is zero when the two planes or lines are the same:

$$e(\mathbf{m}, \mathbf{m}') = \|\mathbf{p}_m - \mathbf{p}_{m'}\|_{\mathbf{A}_{m'}}^2 + \|\mathbf{d}_m - \mathbf{d}_{m'}\|^2 \quad (5.5)$$

While, for the distance between a line and a plane, we consider that a line lies on a plane if: the point of the line lies on the plane, their direction vectors are orthogonal. The first requirement is satisfied when Eq. (5.4) evaluates to 0. To capture the orthogonality constraint expressed by the second requirement we add to the metric an additional semi-positive term that is zero when the two directions are orthogonal:

$$e(\mathbf{m}, \mathbf{m}') = \|\mathbf{p}_m - \mathbf{p}_{m'}\|_{\mathbf{A}_{m'}}^2 + \|\mathbf{d}_m^\top \mathbf{d}_{m'}\|^2 = 0 \quad (5.6)$$

Having considered how to measure the distance between all possible combinations of matchables, we want to have a unique formulation for all of them. This will be useful in the remainder, as we will derive the registration procedure only once for all possible types of constraint.

To this end, we rewrite the difference between two matchables as the following 7D vector:

$$\mathbf{e}(\mathbf{m}, \mathbf{m}') = \begin{pmatrix} e_p \\ e_d \\ e_o \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{m'}^\top (\mathbf{p}_m - \mathbf{p}_{m'}) \\ \mathbf{d}_m - \mathbf{d}_{m'} \\ \mathbf{d}_m^\top \mathbf{d}_{m'} \end{pmatrix} \quad (5.7)$$

A distance  $e(\mathbf{m}, \mathbf{m}')$  between matchables is a non-negative scalar computed from the difference vector. If the distance is zero, the constraint between the two matchables is satisfied. To compute the distance, we employ adapted  $\Omega$ -norm (*i.e.*  $\|\mathbf{v}\|_\Omega = \mathbf{v}^\top \Omega \mathbf{v}$ ) to difference vector:

		$\mathbf{m}'$		
		point	line	plane
$\mathbf{m}$	point	$\Omega_p = \mathbf{I}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 0$	$\Omega_p = \Omega_{\mathbf{m}'}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 0$	$\Omega_p = \Omega_{\mathbf{m}'}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 0$
	line	$\Omega_p = \Omega_{\mathbf{m}}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 0$	$\Omega_p = \Omega_{\mathbf{m}'}$ $\Omega_d = \mathbf{I}$ $\Omega_o = 0$	$\Omega_p = \Omega_{\mathbf{m}'}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 1$
	plane	$\Omega_p = \Omega_{\mathbf{m}}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 0$	$\Omega_p = \Omega_{\mathbf{m}}$ $\Omega_d = \mathbf{0}$ $\Omega_o = 1$	$\Omega_p = \Omega_{\mathbf{m}'}$ $\Omega_d = \mathbf{I}$ $\Omega_o = 0$

Table 5.2. Information Matrix  $\Omega(\mathbf{m}, \mathbf{m}')$  for each possible pair of matchables.

$$\begin{aligned}
e(\mathbf{m}, \mathbf{m}') &= \|\mathbf{e}(\mathbf{m}, \mathbf{m}')\|_{\Omega(\mathbf{m}, \mathbf{m}')}^2 \\
&= \mathbf{e}(\mathbf{m}, \mathbf{m}')^\top \Omega(\mathbf{m}, \mathbf{m}') \mathbf{e}(\mathbf{m}, \mathbf{m}').
\end{aligned} \tag{5.8}$$

The information matrix  $\Omega(\mathbf{m}, \mathbf{m}') \in \Re^{7 \times 7}$  activates the appropriate components of the difference vector during the minimization, based on the type of constraint, according to Tab. 5.2. In particular, we enforce the following block diagonal structure for  $\Omega(\mathbf{m}, \mathbf{m}')$ :

$$\Omega(\mathbf{m}, \mathbf{m}') = \begin{pmatrix} \Omega_p & 0 & 0 \\ 0 & \Omega_d & 0 \\ 0 & 0 & \Omega_o \end{pmatrix} \tag{5.9}$$

With this formulation, the generic distance between two matchables is computed as:

$$e(\mathbf{m}, \mathbf{m}') = \|\mathbf{e}(\mathbf{m}, \mathbf{m}')\|_{\Omega(\mathbf{m}, \mathbf{m}')}^2 \tag{5.10}$$

$$\begin{aligned}
&= \mathbf{e}(\mathbf{m}, \mathbf{m}')^\top \Omega(\mathbf{m}, \mathbf{m}') \mathbf{e}(\mathbf{m}, \mathbf{m}') \\
&= \|\mathbf{e}_p\|_{\Omega_p}^2 + \|\mathbf{e}_d\|_{\Omega_d}^2 + \|e_o\|_{\Omega_o}^2
\end{aligned} \tag{5.11}$$

Consequently, the lower  $e(\mathbf{m}, \mathbf{m}')$  is, the “closer” the two primitives are.

#### 5.1.4 Registration

Having defined how to transform the matchables, and a way to compute how “far” is a constraint from being satisfied, we seek for the optimal transformation  $\mathbf{X}^*$  that better aligns two scenes:

$$\begin{aligned}
\mathbf{X}^* &= \operatorname{argmin}_{\mathbf{X}} \sum_k e(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k) \\
&= \operatorname{argmin}_{\mathbf{X}} \sum_k \left\| \mathbf{e}(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k)^T \right\|_{\Omega(\mathbf{m}_k, \mathbf{m}'_k)}^2
\end{aligned} \tag{5.12}$$

for each pair of matchables  $\langle \mathbf{m}_k, \mathbf{m}'_k \rangle_{1:K}$  between the “moving” and the “fixed” scene.

In Chapter 3 we have seen that a standard way to minimize the above equation is through the Gauss-Newton iterative optimization of Alg. 2. In this context, we propose two different schemes for the minimization problem: a non-linear iterative solver and a direct solver. The non-linear solution is obtained by iteratively performing the optimization on a local parameterization of the perturbations. The direct solution does not require an initial guess of the transform, and finds the minimum in just one iteration. It is obtained by relaxing the constraint on the rotation matrix of the transform, and then recovering the orthonormality of the solution through Singular Value Decomposition (SVD).

Consequently, we specify for both cases, *iterative* and *direct*, the specific representation of the state space  $\mathbf{X}$ , the perturbation vector  $\Delta \mathbf{x}$  and the  $\boxplus$  operator to perform the update. For completeness, we report the derivation of the Jacobian matrix of line 12 of Alg. 2.

### Iterative Solver

We represent the perturbation vector  $\Delta \mathbf{x} \in \mathfrak{R}^6$  as the following vector:

$$\Delta \mathbf{x} := \underbrace{(\Delta x \ \Delta y \ \Delta z)}_{\Delta \mathbf{t}} \underbrace{(\Delta \alpha_x \ \Delta \alpha_y \ \Delta \alpha_z)}_{\Delta \alpha}^T, \tag{5.13}$$

where  $\Delta \mathbf{t}$  is the translational part, while  $\Delta \alpha$  embeds the rotations around the  $x$ ,  $y$  and  $z$  axes. To apply this perturbation to  $\mathbf{X}$  we first convert  $\Delta \mathbf{x}$  into a rotation matrix and a translation vector through the  $\text{v2t}$  function, and then we multiply the homogeneous transforms:

$$\Delta \mathbf{X} = \text{v2t}(\Delta \mathbf{x}) \tag{5.14}$$

$$= \begin{pmatrix} \mathbf{R}_x(\Delta \alpha_x) \mathbf{R}_y(\Delta \alpha_y) \mathbf{R}_z(\Delta \alpha_z) & \Delta \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

$$\mathbf{X} \boxplus \Delta \mathbf{x} := \Delta \mathbf{X} \cdot \mathbf{X} \tag{5.15}$$

We can expand the Jacobian  $\mathbf{J}$  as follows:

$$\begin{aligned}
\mathbf{J} &= \left. \frac{\partial e((\mathbf{X} \boxplus \Delta \mathbf{x})\mathbf{m}, \mathbf{m}')}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0} \\
&= \left. \frac{\partial e(\text{v2t}(\Delta \mathbf{x}) \overbrace{\mathbf{X}\mathbf{m}}^{\tilde{\mathbf{m}}}, \mathbf{m}')}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0}
\end{aligned} \tag{5.16}$$

$$= \underbrace{\left. \frac{\partial e(\mathbf{m}, \mathbf{m}')}{\partial \mathbf{m}} \right|_{\mathbf{m}=\tilde{\mathbf{m}}}}_{\mathbf{J}_e} \underbrace{\left. \frac{\partial \text{v2t}(\Delta \mathbf{x}) \tilde{\mathbf{m}}}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0}}_{\mathbf{J}_x}. \tag{5.17}$$



The calculations leads to the following form for the difference Jacobian  $\mathbf{J}_e$  and the transformation Jacobian  $\mathbf{J}_x$ :

$$\mathbf{J}_e = \begin{pmatrix} \mathbf{R}_{\mathbf{m}'}^\top & -\mathbf{R}_{\mathbf{m}'}^\top [\tilde{\mathbf{p}}_{\mathbf{m}}]_\times \\ \mathbf{0} & -[\tilde{\mathbf{d}}_{\mathbf{m}}]_\times \\ \mathbf{0} & [1 \ 0 \ 0] \mathbf{R}_{\mathbf{m}}^\top \mathbf{R}_{\mathbf{x}}^\top [\mathbf{d}'_{\mathbf{m}}]_\times \end{pmatrix} \quad (5.18)$$

$$\mathbf{J}_x = \begin{pmatrix} \mathbf{I}_{3 \times 3} & -[\tilde{\mathbf{p}}_{\mathbf{m}}]_\times \\ \mathbf{0} & -[\tilde{\mathbf{d}}_{\mathbf{m}}]_\times \end{pmatrix}. \quad (5.19)$$

Here,  $[\mathbf{a}]_\times$  denotes the skew symmetric matrix obtained from a 3D vector  $\mathbf{a}$ , while  $\tilde{\mathbf{p}}_{\mathbf{m}}$  and  $\tilde{\mathbf{d}}_{\mathbf{m}}$  are respectively the Euclidean and direction components of  $\tilde{\mathbf{m}} = \mathbf{X} \cdot \mathbf{m}$ .

### Direct Solver

For the direct solution of the Gauss-Newton algorithm, we consider a perturbation  $\Delta \mathbf{x} \in \mathfrak{R}^{12}$  composed as follows

$$\Delta \mathbf{x} = \left( \Delta \mathbf{t}^T \ \Delta \mathbf{r}_1^T \ \Delta \mathbf{r}_2^T \ \Delta \mathbf{r}_3^T \right)^T \quad (5.20)$$

where  $\Delta \mathbf{r}_i^T$  stands for the  $i^{th}$  column of a rotation matrix  $\Delta \mathbf{R}$

$$\Delta \mathbf{R} = \begin{pmatrix} \Delta \mathbf{r}_1^T \\ \Delta \mathbf{r}_2^T \\ \Delta \mathbf{r}_3^T \end{pmatrix}. \quad (5.21)$$

We define the  $\boxplus$  operator as:

$$\mathbf{X} \boxplus \Delta \mathbf{x} := \begin{pmatrix} \mathbf{R} + \Delta \mathbf{R} & \mathbf{t} + \Delta \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (5.22)$$

Notice that during the solution we do not enforce the orthonormality of the rotation matrices involved. To lessen this problem, after applying the perturbation, we recondition the rotation matrix of  $\mathbf{X}$  through SVD decomposition as follows:

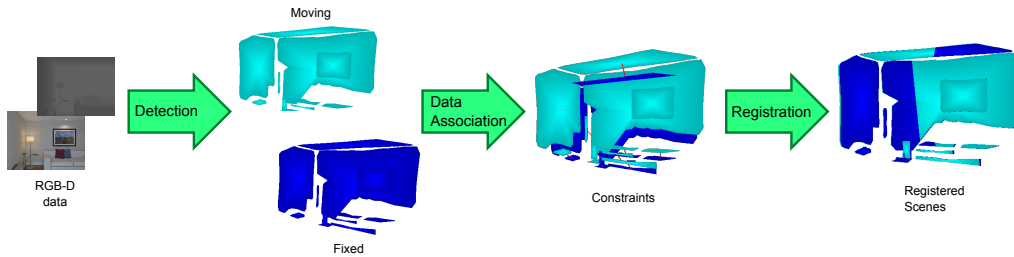
$$\mathbf{U}\mathbf{S}\mathbf{V} = \mathbf{R} \quad (5.23)$$

$$\mathbf{R}' = \mathbf{U}\mathbf{V}. \quad (5.24)$$

The above procedure provides us with the ‘‘closest’’ rotation matrix to the original  $\mathbf{R}$ . If  $\mathbf{S}$  is close to the identity, the SVD approximation is good, otherwise the direct solver provides a suboptimal solution. This typically occurs when several outliers are present in the constraints. Similar to the iterative case, the Jacobian can be computed from Eq. (5.19), and using the chain rule we need to rewrite only the transformation Jacobian  $\mathbf{J}_x$ :

$$\mathbf{J}_x = \begin{pmatrix} \mathbf{I}_{3 \times 3} & \mathbf{p}_{\mathbf{m}}^T & & \\ & \mathbf{p}_{\mathbf{m}}^T & & \\ & & \mathbf{p}_{\mathbf{m}}^T & \\ \mathbf{0} & \mathbf{d}_{\mathbf{m}}^T & \mathbf{d}_{\mathbf{m}}^T & \mathbf{d}_{\mathbf{m}}^T \end{pmatrix}. \quad (5.25)$$

Since in the direct case the Jacobian does not depend on the transformation, it does not change during the iterations. Accordingly, a solution can be found in a single iteration.



**Figure 5.2.** Front-end. When a new pair of images is available, we extract a set of matchables from raw data using the strategy outlined in Sec. 5.2.1, here we show only planes for more clarity. Subsequently, we find corresponding matchables (here, linked by a red line) between the newly generated scene and the “fixed” scene with the methodology in Sec. 5.2.2. Finally, a minimization is conducted to find the transform that best aligns the two scenes according to Sec. 5.1.4.

## 5.2 Front-End

To validate the representation and the solvers proposed in Sec. 5.1 on real data we realized a front-end based on RGB-D data. The processing pipeline, as shown in Fig. 5.2, is divided in three steps: detection, data association and registration.

### 5.2.1 Detecting Matchables from RGB-D data

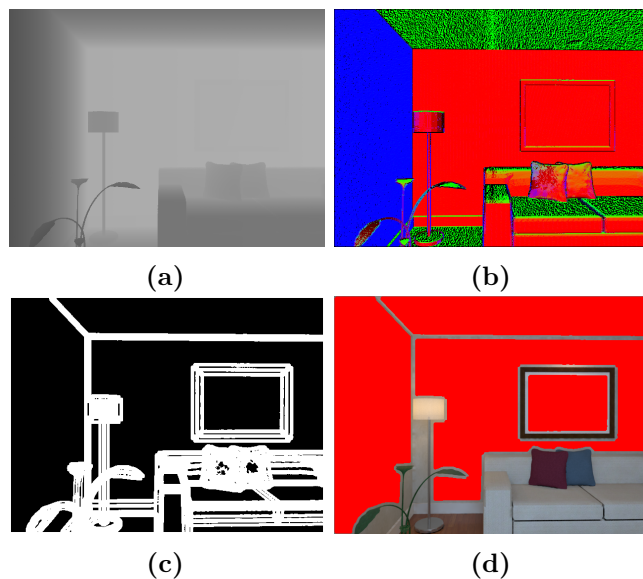
In Sec. 2.1.3 we presented different methods to detect features in sensors data. Arguably, the most common technique is to extract corners from intensity images to obtain interest points. At the same time, most man-made objects are made of flat surfaces, this results in image edges that form straight lines. In a textureless scene, where a point feature detector is likely to fail, there are still good chances to detect lines.

Image based feature detectors do not account for the geometry of the scene and rely solely on the intensity channel. For this reason, their main limitation is a performance drop in detection under varying or severe lighting conditions. On the other hand, plane detection from range data depends on the 3D structure of the scene and is immune to light changes. However, plane detection depends on the estimation of geometric features (*i.e.* surface normals and curvature) which may be severely limited by noisy and non-uniform data returned by range sensors.

Since our representation comprehends all these geometric entities, we can combine different detection techniques to compensate for their singular shortcomings. In the remainder we present the details on the detection of these primitives in our system.

#### Points and Lines

Points and lines are extracted in image coordinates using the intensity channel. In particular we use the FAST detector [125] to find salient pixels in the image, and we add to the scene the corresponding 3D point retrieved from the depth channel. Lines are found on the intensity image using the Line Segment Detector introduced in [167]. Each line in the image is back-projected by considering the 3D points at the extrema. To lessen the effect of spurious detections, we reject all lines that are



**Figure 5.3.** Planes extraction. From left to right and top to bottom: (a) input depth image, (b) estimated surface normals, (c) connected regions (in black), (d) detected planes (in red).

too short or do not have sufficient support from the depth data. As an additional cue for points and lines matching, we label each point with a BRIEF descriptor [12] computed at the keypoint location. Similarly, we add to each line matchable a descriptor computed according to [176].

### Planes

We first generate an organized 3D point cloud from the depth image and compute the surface normals for each point according to [30]. Subsequently, we mark each point characterized by a significant local variation of depth or normal. After this procedure, each pixel is classified as belonging to a continuous region or not and we recover the connected regions by visiting the image according to an 8 neighbor connectivity (see Fig. 5.3). For each of these regions that has a minimum number of points, we fit a plane. If the residual error of the fitted plane is below a threshold, we add a new plane to the scene. We repeat this procedure until no planes can be generated.

#### 5.2.2 Data Association

The set of matchables detected in the current RGB-D frame constitutes the moving scene  $\mathcal{S} = \{\mathbf{m}_{1:N}\}$ . While, the set of matchables previously detected and registered in a global reference frame forms the fixed scene  $\mathcal{S}' = \{\mathbf{m}'_{1:N'}\}$ . The goal of the *data association* module is to find corresponding elements between the two sets. Then, each correspondence will result in a constraint for the solver.

Given the formalism presented in Sec. 5.1.1, we are able to find two types of constraints: homogeneous and heterogeneous. A homogeneous constraint arises when we re-observe the same matchable, *e.g.* a point in the moving scene *matches* a point

in the fixed scene. While, a heterogeneous constraint derives from a correspondence between primitives of different types, such as a line in the moving scene *lies* on a plane in the fixed scene.

We perform the constraints search in two steps, *i.e.* we first look for matchables of the same type and then we perform a hybrid match search.

### Homogeneous Constraints

We designed a data association schema based on a Nearest Neighbor Search. Given the fixed scene  $\mathcal{S}'$ , the moving scene  $\mathcal{S}$ , and the current estimate  $\mathbf{X}$ , for each matchable  $\mathbf{m}$  we want to find the matchable  $\mathbf{m}'^*$  such that the error in Eq. (5.10) is minimized:

$$\mathbf{m}'^* = \underset{\mathbf{m}' \in \mathcal{S}}{\operatorname{argmin}} e(\mathbf{X}\mathbf{m}, \mathbf{m}'). \quad (5.26)$$

For efficiency, we perform an approximate search, organizing the matchables of the fixed scene  $\mathcal{S}'$  in three KD-trees: one for each primitive. We perform an additional culling to remove *bad* associations. Matched planes are discarded if their distance is bigger than a certain threshold, while for points and lines we reject those associations whose descriptors are not similar.

### Heterogeneous Constraints

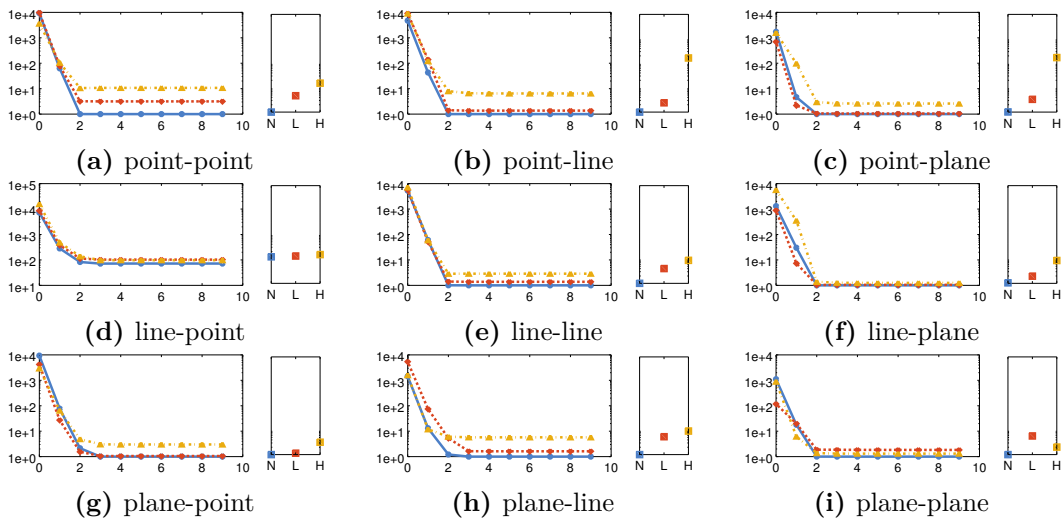
Once performed the homogeneous search, we use the unassociated matchables of the moving scene  $\mathbf{m}$  to perform a brute force search with the matchables of different type in the fixed scene. We evaluate the hybrid matchables distance as in Eq. (5.10), and we discard the constraints with error higher than a specified threshold  $t_{cross}$ .

## 5.3 Experimental Evaluation

In this section we present a set of experiments aiming at evaluating the performance of both the iterative and the direct version of the solver proposed in Section 5.1. To this extent, we used a synthetically generated scene with known correspondences under different levels of noise and we analyzed the convergence behavior of the solver under different constraints. Section 5.3.1 reports the outcome of this set of experiments. Subsequently, in Section 5.3.2 we report the results of a comparative evaluation that involves a full registration pipeline built on top of our solver and other state-of-the-art approaches. The experiments have been conducted on standard benchmarking datasets.

### 5.3.1 Synthetic Data

We designed a first experiment with synthetic data to validate our method presented in Section 5.1. We isolate the effects of the front-end and analyze the behavior of each type of constraint on both iterative and direct solver. To this end, for each type of constraint, we generated two noise-free scenes with 10 matchables of each type, such that the corresponding constraints were fully satisfied.



**Figure 5.4.** Synthetic Experiments - Error Evolution. In *solid blue* is shown the evolution of the error without noise, while in *dashed red* is reported the low-noise case and in *point-dashed yellow* the high-noise case. For each constraint, the left plot reports the Iterative Solver evolution, while the right plot shows the error after one Direct Solver iteration.

In Fig. 5.4 we report the error evolution of the iterative solver for 10 iterations, while for the direct solver we present the result after one iteration. To estimate the effect of noise in the measurements we repeated the previous experiment by adding varying levels of noise to the matchables in one scene. The corresponding curves are marked as *N: no-noise*, *L: low-noise* [ $\sigma_t^2 \sim 0.05$ ,  $\sigma_r^2 \sim 0.025$ ] and *H: high-noise* [ $\sigma_t^2 \sim 0.1$ ,  $\sigma_r^2 \sim 0.05$ ].

For the same type of constraint and level of noise both solvers operate on the same data and start from the same initial guess [ $t_x = 0.3$ ,  $t_y = -0.8$ ,  $t_z = 0.6$ ,  $q_x = 0.5$ ,  $q_y = -0.5$ ,  $q_z = 0.5$ ]. As evident from Fig. 5.4 both iterative and direct variants of our solver converge close to the optimum. As expected the iterative solver is less sensitive to the measurement noise, at a cost of multiple iterations. Conversely the direct solver should be preferred in low noise situations since it finds the optimum in just one step.

### 5.3.2 Simulated and Real Data

We compared the position tracking performances of our approach and three other state-of-the-art methods, namely: Normal ICP (NICP) [138], Dense Visual Odometry (DVO) [74] and Multi-Cue Photometric Registration (MPR) [30]. We used the author’s open source implementations<sup>1</sup>. The comparison has been conducted on the following publicly available datasets with ground truth:

- ICL-NUIM dataset [55], simulated RGB-D measurements in indoor scenarios.

<sup>1</sup>Source: [https://github.com/tum-vision/dvo\\_slam](https://github.com/tum-vision/dvo_slam) (V.O. only)  
Source: <https://srrg.gitlab.io/nicp.html>  
Source: <https://srrg.gitlab.io/mpr.html>

- TUM benchmark suite [146], acquired with a Kinect v1 sensor in office-like environments.

For all the presented experiments we provide the achieved accuracy in terms of relative pose error (RPE), computed with the evaluation script provided by the TUM benchmark suite. For both datasets, we set  $t_{cross} = 0.01$  as threshold to discard heterogeneous associations, as described in Sec. 5.2.2.

In Tab. 5.3 we report the results of each approach in terms of relative translational and rotational pose error per second on the ICL-NUIM sequences. The presented results show that our approach achieves accuracy comparable with state-of-the-art methods. To highlight the effects of using high-level primitives we present two variants of our approach, denoted as *SA-PT* and *SA-SHA*. *SA-PT* considers only points, while *SA-SHA* uses all primitives generated by the front end. *SA-PT* and *SA-SHA* does not present significant differences except in the first two sequences, where the camera is repeatedly pointed towards textureless walls, and salient features are poorly detected, causing an observable weakness in the *SA-PT* performances. On the other hand, the *SA-SHA* version takes advantage of the high-level geometric primitives. To provide the reader with an estimate of the memory occupancy of our method, we also report the number of matchables for our approach and the point cloud size of [138] in the final reconstructed scenes.

We repeated the same procedure on eight sequences of the TUM benchmark suite. The data have been acquired with a Kinect v1 in office-like environments. Despite the simplistic front-end procedure we used in this work to extract matchables, our approach performs well on real data, as reported in Tab. 5.4. *SA-SHA* outperforms *SA-PT* in all the sequences, except when no structure is present (fr3/nostr-text-near-loop), where the two versions perform similarly. This encourages to further improve the front-end part to perform a more robust registration.

	MPR		DVO		NICP		SA-PT		SA-SHA	
	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]
lr/tr0	0.0182	0.7055	<b>0.0063</b>	<b>0.5347</b>	<b>0.0063</b>	0.5403	0.0121	0.6466	0.0077	0.5499
					{2041773}				{956 / 82 / 14}	
lr/tr1	0.0216	0.6307	0.0019	0.4773	<b>0.0014</b>	<b>0.4751</b>	0.0134	0.5802	0.0042	0.4857
					{1357503}				{608 / 75 / 16}	
lr/tr2	0.0193	0.6366	<b>0.0046</b>	0.4983	0.0047	0.4948	0.0071	0.5667	0.0053	<b>0.3843</b>
					{1865157}				{1180 / 69 / 13}	
lr/tr3	0.0218	0.6351	<b>0.0055</b>	0.3607	0.0065	0.3907	0.0096	0.6553	0.0059	<b>0.3512</b>
					{2997922}				{981 / 65 / 10}	
tr0	0.0243	0.6464	<b>0.0064</b>	<b>0.4967</b>	0.0068	0.4988	0.0104	0.5254	0.0094	0.5159
					{2140708}				{1149 / 85 / 13}	
tr1	0.0272	0.7173	0.0045	0.4834	<b>0.0043</b>	<b>0.4783</b>	0.0069	0.5223	0.0057	0.4918
					{1654631}				{999 / 105 / 17}	
tr2	0.0309	0.7469	0.0053	<b>0.5068</b>	<b>0.0052</b>	0.5078	0.0094	0.5472	0.0071	0.5146
					{1895441}				{890 / 68 / 15}	
tr3	0.0223	0.5733	<b>0.0037</b>	0.3567	0.0039	<b>0.3533</b>	0.0056	0.3685	0.0052	0.3639
					{2300953}				{1257 / 145 / 16}	

Table 5.3. ICL-NUIM - Relative Pose Error.

	MPR		DVO		NICP		SA-PT		SA-SHA	
	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]	[m/s]	[rad/s]
fr1/desk	0.0617	3.3369	<b>0.0487</b>	2.8261	0.1123	8.6928	0.0919	5.0505	0.0888	<b>2.6464</b>
					{15488164}				{282 / 29 / 2}	
fr1/desk2	0.0921	5.1575	<b>0.0752</b>	<b>4.4137</b>	0.1989	9.9035	0.1565	6.1042	0.0974	5.9066
					{18325912}				{295 / 33 / 1}	
fr2/desk	0.0364	1.6511	0.0377	1.5691	0.1032	2.8895	0.0236	0.9921	<b>0.0206</b>	<b>0.8661</b>
					{17430749}				{7934 / 59 / 2}	
fr2/person	0.0478	1.4512	0.0332	0.9791	0.3479	13.8372	0.0245	0.8820	<b>0.0224</b>	<b>0.7693</b>
					{20887437}				{10146 / 45 / 4}	
fr3/long-household	<b>0.0262</b>	1.2773	0.0441	1.4096	0.1412	4.8721	0.0354	1.4398	0.0274	<b>1.1352</b>
					{24204397}				{3849 / 93 / 10}	
fr3/nostr-text-near-loop	0.2505	7.6629	0.0249	1.0935	0.3179	8.9081	<b>0.0211</b>	<b>0.9506</b>	0.0227	1.0219
					{893384}				{2058 / 154 / 1}	
fr3/str-text-far	0.0413	1.2862	0.0943	2.4563	0.1002	1.4571	0.0240	0.6646	<b>0.0228</b>	<b>0.6546</b>
					{3133948}				{713 / 107 / 5}	
fr3/str-text-near	0.0451	2.1518	0.0364	1.9501	0.0736	2.3584	0.0376	1.6553	<b>0.0282</b>	<b>1.2725</b>
					{4480124}				{1223 / 26 / 2}	

Table 5.4. TUM - Relative Pose Error.





## Chapter 6

# Unifying Global Optimization Algorithms

Sensor range and field-of-view limits cause each frame to contain only a partial view of the environment. To observe it all, the robot has to move within its workspace so that, as described in Chapter 4, the map is built incrementally by registering and fusing together the measurements acquired during exploration. During this process inconsistencies may arise due to the accumulation of a drift.

In this context, a crucial requirement is to maintain a globally consistent map over time. For this reason, modern SLAM systems are typically made of four components: a sensor tracker, to estimate the current pose of the sensor (camera, laser, etc...) along the trajectory; a local map manager, to gather a set of measurements from a chunk of trajectory that relate to the same portion of the environment; a loop detector, to recognize if a place has been revisited and a global optimizer to determine the state configuration that better fits the measurements.

In the last decade, graph-based approaches have shown to be effective solutions to the full SLAM problem and techniques for both constructing the graph (front-end) and carrying on the optimization (back-end) have been proposed (Sec. 2.2). The majority of these approaches consider simple, or low-level, geometric primitives in their formulation, like points. However, as shown in Chapter 5, higher level primitives can improve registration results thanks to their greater descriptiveness.

Both for filtering and smoothing approaches, the use of lines and planes has been investigated to improve estimation accuracy. In this chapter, we adopt the representation introduced in Sec. 5.1.1 that is able to describe different types of primitive with a single parameterization and can capture both homogeneous and heterogeneous constraints between them. In this way, it is possible to recover, in a globally consistent way, the sensor poses along with a map representation of the environment made of points, lines and planes.

Based on this representation, we derive in Sec. 6.1 a global optimization procedure to solve the full SLAM problem. Subsequently, in Sec. 6.2 we present the details of the front-end we implemented to test our method on real data. Finally, an experimental evaluation based on both synthetic and real data is reported in Sec. 6.3.

## 6.1 Multi-Primitive Registration

In this section, we derive a generalized version of the Multi-Point Registration algorithm of Sec. 3.3.1 to obtain a globally optimal reconstruction of the sensor poses and the scene made of primitives.

The input to this algorithm is a factor graph (see Sec. 3.3.2) that encodes an initial guess of the sensor poses and the primitives positions along with a set of constraints (or edges). Each edge may specify: (a) from which pose a primitive has been observed and (b) a relative transform between two poses.

Based on the parameterization introduced in Sec. 5.1.1, we first define the state space of the optimization problem and how to apply a perturbation to it. With this basic operation defined, we show how to compute an error between the predicted and the measured matchables. Finally, we derive a Gauss-Newton solver capable of finding the state configuration that better explains the measurements.

### 6.1.1 State

Similar to Sec. 3.3, we assume to have a sensor capable of perceiving distinguishable geometric primitives that is used to explore an unknown environment. Therefore, the state  $\mathbf{X}$  of our problem will be composed by the set of poses at which new measurements have been acquired and the set of *matchables* seen along the trajectory:

$$\mathbf{X} = \{\mathbf{X}_r^{[1]}, \dots, \mathbf{X}_r^{[N]}, \mathbf{X}_m^{[1]}, \dots, \mathbf{X}_m^{[M]}\} \quad (6.1)$$

with:

- sensor pose  $\mathbf{X}_r^{[n]} \in SE(3)$
- matchable  $\mathbf{X}_m^{[m]}$  as in Eq. (5.2).

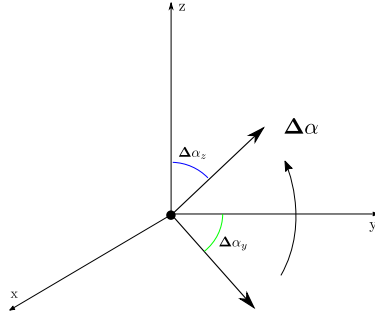
The goal is to find the best  $\mathbf{X}^*$  that satisfies the factor-graph constraints. As we have seen in Sec. 3.1.2, non-linear optimization techniques work by locally applying a perturbation to the state in order to reach the optimum. For the sensor pose, the perturbation is defined as in Eq. (3.52). While, we define the euclidean perturbation of a matchable as:

$$\Delta \mathbf{m} \in \mathfrak{R}^5 : \Delta \mathbf{m} = \underbrace{(\Delta x \ \Delta y \ \Delta z)}_{\Delta \mathbf{p}} \underbrace{(\Delta \alpha_y \ \Delta \alpha_z)}_{\Delta \alpha}^T, \quad (6.2)$$

where  $\Delta \mathbf{p}$  account for the translation and  $\Delta \alpha$  represents the rotation around the  $y$  and  $z$  axes. The latter is used to perturb the direction of a matchable, see Fig. 6.1. To express this perturbation only two parameters are needed because, intuitively, rotating the vector also around the  $x$  axis has no effect on the direction. Another way to see this, is that a direction can be considered as a point on the unit sphere, thus, it may be expressed in spherical coordinates with just two parameters: the polar and the azimuth angle. It follows that the perturbation is applied to a matchable in the following way:

$$\mathbf{m}' = \mathbf{m} \boxplus \Delta \mathbf{m} \quad (6.3)$$

$$= \begin{pmatrix} \mathbf{p}_m + \Delta \mathbf{p} \\ \mathbf{R}_m \cdot \Delta \mathbf{R} \\ \Omega_m \end{pmatrix} \quad (6.4)$$



**Figure 6.1.** Applying a perturbation to the matchable direction.

where  $\Delta \mathbf{R} = \mathbf{R}_y(\alpha_y) \cdot \mathbf{R}_z(\alpha_z)$  is the composition of the elementary rotations around the  $y$  and  $z$  axes. The reader might notice that it is multiplied to the right of the matchable rotation matrix. In this way, the perturbation is first evaluated in the origin and then added to  $\mathbf{R}_m$ .

Having defined how to perturb a matchable, we can now consider how to apply an increment to the whole state. In particular, the increments  $\Delta \mathbf{x} \in \mathfrak{R}^{6N+5M}$  will be represented by a large vector containing the minimal perturbation for each state variable:

$$\Delta \mathbf{x} = (\Delta \mathbf{x}_r^{[1]\top}, \dots, \Delta \mathbf{x}_r^{[N]\top}, \Delta \mathbf{x}_m^{[1]\top}, \dots, \Delta \mathbf{x}_m^{[M]\top})^\top \quad (6.5)$$

with:

$$\begin{aligned} \Delta \mathbf{x}_r^{[n]\top} &\in \mathfrak{R}^6 : & (6.6) \\ \Delta \mathbf{x}_r^{[n]\top} &= \underbrace{(\Delta x^{[n]} \Delta y^{[n]} \Delta z^{[n]})}_{\Delta \mathbf{t}^{[n]}} \underbrace{(\Delta \alpha_x^{[n]} \Delta \alpha_y^{[n]} \Delta \alpha_z^{[n]})}_{\Delta \alpha^{[n]}} \end{aligned}$$

and  $\Delta \mathbf{x}_m^{[m]\top} \in \mathfrak{R}^5$  as defined in Eq. (6.2). Applying the individual perturbations for each variable block results in:

$$\begin{aligned} \mathbf{X}' &= \mathbf{X} \boxplus \Delta \mathbf{x} & (6.7) \\ \mathbf{X}_r^{[n]'} &= \Delta \mathbf{x}_r^{[n]} \boxplus \mathbf{X}_r^{[n]} \\ &= v2t(\Delta \mathbf{x}_r^{[n]}) \mathbf{X}_r^{[n]} \\ \mathbf{X}_m^{[m]'} &= \Delta \mathbf{x}_m^{[m]} \boxplus \mathbf{X}_m^{[m]} \end{aligned}$$

### 6.1.2 Error

In Sec. 5.1.3, we have shown that the proposed parameterization for a *matchable* allows to define a single error function that can measure the distance between different primitives both of the same type and heterogeneous. In this section, we exploit this representation to define the error between a predicted and a measured matchable. This holds true regardless of the matchables type and allows to derive the Jacobians of the error function only once.

We define the measurement of the matchable  $m$ , performed by the sensor pose  $n$  as:

$$\mathbf{z}^{[n,m]} \in \mathfrak{R}^{15} : \quad (6.8)$$

$$\mathbf{z}^{[n,m]} = \underbrace{(x^{[n,m]}, y^{[n,m]}, z^{[n,m]})}_{\mathbf{p}_m^{[n,m]}} , \underbrace{(r_{1,1}^{[n,m]}, \dots, r_{3,3}^{[n,m]})}_{\text{linearized } \mathbf{R}_m^{[n,m]}} , \underbrace{(\omega_x^{[n,m]}, \omega_y^{[n,m]}, \omega_z^{[n,m]})^\top}_{\Omega_m^{[n,m]} \text{ diagonal}}$$

Given the state  $\mathbf{X}$  and the difference vector defined in Eq. (5.7), the predictions and the error of these measurements are:

$$\mathbf{h}^{[n,m]}(\mathbf{X}) = \mathbf{X}_r^{[n]} \mathbf{X}_m^{[m]} \quad (6.9)$$

$$\mathbf{e}^{[n,m]}(\mathbf{X}) = \mathbf{X}_r^{[n]} \mathbf{X}_m^{[m]} - \mathbf{z}^{[n,m]} \quad (6.10)$$

$$\mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x}) = v2t(\Delta \mathbf{x}_r^{[n]}) \mathbf{X}_r^{[n]} (\Delta \mathbf{x}_m^{[m]} \boxplus \mathbf{X}_m^{[m]}) - \mathbf{z}^{[n,m]} \quad (6.11)$$

The jacobian  $\mathbf{J}$  of the error function between the actual measurement  $m$  and the one predicted from pose  $n$  is:

$$\begin{aligned} \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} &= \\ &= (\dots \mathbf{0}_{7 \times 6} \dots \underbrace{\frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_r^{[n]}}}_{\mathbf{J}_r^{n,m}} \dots \mathbf{0}_{7 \times 6} \dots \\ &\quad \dots \mathbf{0}_{7 \times 6} \dots \underbrace{\frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \Delta \mathbf{x})}{\partial \Delta \mathbf{x}_m^{[m]}}}_{\mathbf{J}_m^{n,m}} \dots \mathbf{0}_{7 \times 6} \dots). \end{aligned} \quad (6.12)$$

with:

$$\mathbf{J}_r^{n,m} = \begin{pmatrix} \frac{\partial \mathbf{e}_p^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} \\ \frac{\partial \mathbf{e}_d^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} \\ \frac{\partial \mathbf{e}_o^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} \end{pmatrix} \quad (6.13)$$

that can be decomposed according to the difference vector components:

$$\frac{\partial \mathbf{e}_p^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} = \left[ \mathbf{R}_{m'}^\top \quad - \mathbf{R}_{m'}^\top S(\mathbf{R}_x \mathbf{p}_m + \mathbf{t}_x) \right] \quad (6.14)$$

$$\frac{\partial \mathbf{e}_d^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} = \left[ \mathbf{0}_{3 \times 3} \quad - S \left( \mathbf{R}_x \mathbf{R}_m \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \right] \quad (6.15)$$

$$\frac{\partial \mathbf{e}_o^{[n,m]}}{\partial \Delta \mathbf{x}_r^{[n]}} = \left[ \mathbf{0}_{1 \times 3} \quad [1 \ 0 \ 0] \mathbf{R}_x^\top \mathbf{R}_m^\top [\mathbf{R}_{m'} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}]_x \right] \quad (6.16)$$

and:

$$\mathbf{J}_m^{n,m} = \begin{pmatrix} \frac{\partial \mathbf{e}_p^{[n,m]}}{\partial \Delta \mathbf{x}_m^{[m]}} \\ \frac{\partial \mathbf{e}_d^{[n,m]}}{\partial \Delta \mathbf{x}_m^{[m]}} \\ \frac{\partial \mathbf{e}_o^{[n,m]}}{\partial \Delta \mathbf{x}_m^{[m]}} \end{pmatrix} \quad (6.17)$$

which can be expanded as:

$$\frac{\partial \mathbf{e}_p^{[n,m]}}{\partial \Delta \mathbf{x}_m^n} = \begin{bmatrix} -\mathbf{R}_{m'}^\top & -F(\mathbf{R}_{m'}^\top(\mathbf{R}_x \mathbf{p}_m + \mathbf{t}_x) - \mathbf{p}_{m'}) \end{bmatrix} \quad (6.18)$$

$$\frac{\partial \mathbf{e}_d^{[n,m]}}{\partial \Delta \mathbf{x}_r^n} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{R}_{m'} F \left( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \end{bmatrix} \quad (6.19)$$

$$\frac{\partial \mathbf{e}_o^{[n,m]}}{\partial \Delta \mathbf{x}_r^n} = \begin{bmatrix} \mathbf{0}_{1 \times 3} & -[1 \ 0 \ 0] \mathbf{R}_m^\top \mathbf{R}_x^\top \mathbf{R}_{m'} F \left( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \end{bmatrix}. \quad (6.20)$$

where we considered only the last two components of the skew-symmetric matrices involved in the derivations:

$$\frac{\Delta \mathbf{R}(\alpha) \mathbf{v}}{\partial \alpha} = -[\mathbf{v}]_x \cdot \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6.21)$$

$$= \underbrace{-F(\mathbf{v})}_{\text{clipped skew}}. \quad (6.22)$$

## 6.2 Front-End

So far we described an optimization problem and followed the methodology of Sec. 3.1.2 to derive an algorithm capable of solving it (back-end). This allows to solve the so-called *graph-based* formulation of SLAM. In order to apply this algorithm on real-world problems, we need a system to collect data and setup the optimization properly (front-end).

In the remainder of this section, we will present some details on the implementation of our front-end. As visible from Fig. 6.2, it has the task of building the factor graph and it comprises a pose tracker, Sec. 6.2.1, and a loop-detector, Sec. 6.2.2.

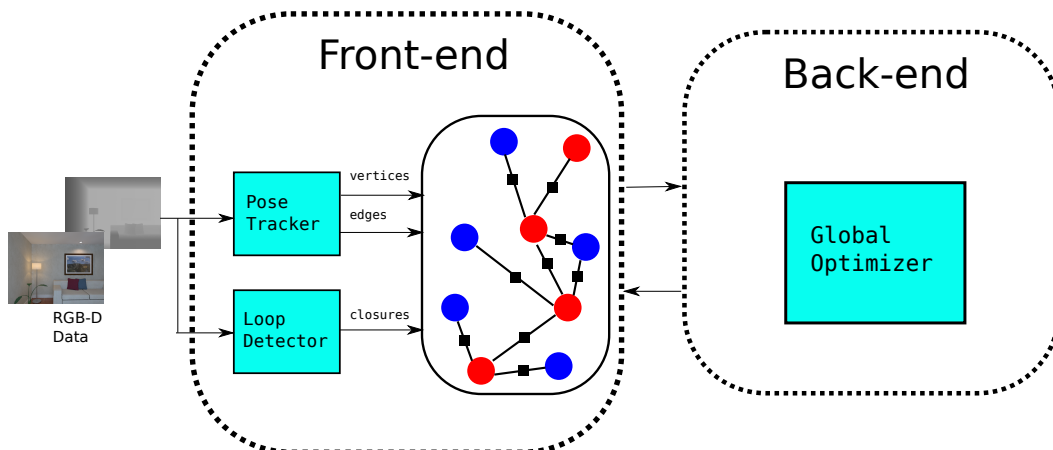


Figure 6.2. Front-end for global optimization.

### 6.2.1 Pose Tracker

To build a factor graph we need three processing modules, namely:

**Matchable Detector:** it is in charge of extracting points, lines and planes from sensor output. It returns the set of matchables  $M = \{\mathbf{m}_i\}$  currently observed by the sensor.

**Correspondence Finder:** it is used to perform *data association* between the current scene returned by the detector and the global map built so far.

**Scene Aligner:** it takes as input the *matches* returned by the correspondence finder and estimates the sensor pose.

Leveraging on the processing pipeline of Sec. 5.2, when a new RGB-D frame is available we are able to estimate the current camera pose. Consequently, we are ready to update the factor graph.

To do so, we add a pose vertex to the graph with the current transform estimate, expressed in the global reference frame, and we link it to the previous pose vertex through a pose-pose edge. Then, we process the observed matchables. To prevent the graph from containing spurious landmarks, we implemented a filtering technique based on the matchables “age”. When we observe a new landmark, its age is initialized to 1. Then, we count how many times it is re-observed, according to the output of the data association. We add a matchable vertex to the graph only if its age is higher than a certain threshold and we link it to the current pose with a pose-matchable edge. For future observations we add further pose-matchable edges. The process is described in Algorithm 4.

### 6.2.2 Loop Detector

In parallel, we process the RGB image for Visual Place Recognition. That is, for a new frame  $f_i$  we first extract  $n_k$  keypoints with the FAST detector and compute a BRIEF descriptor for each of them. Then, we use an Hamming distance-embedding Binary Search Tree (HBST) [133] to find matches between previous frames. The tree returns a vector containing a score  $n_s$  for each past frame, based on the number of matching descriptors. According to this score, we get the best matching frame  $f_j$  and we keep it if it satisfies the following criteria:

- $i - j > \alpha_{th}$
- $n_s > \gamma_{th}$
- $n_s/n_k > \beta_{th}$

with  $\alpha_{th}$ ,  $\beta_{th}$  and  $\gamma_{th}$  arbitrary thresholds. If the previous check is passed, we want to find an alignment between the two frames. To this end, we run the iterative solver of Sec. 5.1.4 using the correspondences returned by the HBST and we encode the estimated transform in a pose-pose edge.

Subsequently, the relative transform is given as initial guess to the KD-Tree for performing data association between scene  $i$  and scene  $j$ . In this way, it is possible to check if there are duplicated vertices in the graph and merge them accordingly.

**Require:** Matchables set:  $M = \{m_i\}$   
**Ensure:** Set of vertices  $V$  and edges  $E$   
to insert into the graph.

- 1: **for**  $i = 1 : N$  **do**
- 2:    $\text{association\_id} \leftarrow m_i.\text{associationId}()$ ; *//retrieve the association identifier*
- 3:   *//CASE 1: un-associated*
- 4:   **if**  $\text{association\_id} < 0$  **then**
- 5:      $m_i.\text{age}() = 1$ ; *//seen for the first time*
- 6:     **continue**;
- 7:   **end if**
- 8:    $m_{\text{ass}} \leftarrow m_i.\text{association}()$ ; *//we found an association*
- 9:    $m_i.\text{age}() \leftarrow m_{\text{ass}}.\text{age}() + 1$ ;
- 10:   *//CASE 2: in graph*
- 11:   **if**  $m_{\text{ass}}.\text{graphId}() \neq -1$  **then**
- 12:      $e \leftarrow \text{generateMatchableMeasurement}(m_i)$ ;
- 13:      $E.\text{insert}(e)$ ; *//add edge to output*
- 14:     **continue**;
- 15:   **end if**
- 16:   *//CASE 3: re-observed but not in graph*
- 17:   **if**  $\text{promotedInGraph}(m_i)$  **then**
- 18:      $v \leftarrow \text{generateMatchableVertex}(m_i)$ ;
- 19:      $e \leftarrow \text{generateMatchableMeasurement}(m_i)$ ;
- 20:      $V.\text{insert}(v)$ ; *//add vertex to output*
- 21:     *//CASE 4: under-age*
- 22:   **else**
- 23:     *//do nothing (and just update stats)*
- 24:   **end if**
- 25: **end for**

**Algorithm 4:** Graph Update for each new matchables scene

## 6.3 Experimental Evaluation

In this section we present two sets of experiments in order to evaluate the performance of the optimization algorithm presented in this chapter. First we analyzed the behavior of our solver on synthetic data. Details on the proposed setting and convergence results are reported in Sec. 6.3.1. Subsequently, we tested the same solver on real data employing the front-end illustrated in Sec. 6.2. The experiments were conducted on a standard benchmarking dataset and are described in Sec. 6.3.2.

### 6.3.1 Synthetic Experiments

To generate the synthetic data for this set of experiments, we implemented a “world simulator”. This allows to: (1) build an environment made of points, lines and planes, (2) generate a motion trajectory for the matchable sensor and (3) collect the virtual measurements acquired along the trajectory.

	$\mathcal{N}_t(0, \Sigma_t)[m]$	$\mathcal{N}_R(0, \Sigma_R)[rad]$	$\mathcal{N}_p(0, \Sigma_p)[m]$	$\mathcal{N}_d(0, \Sigma_d)[rad]$
<b>low-noise</b>	[0.01 0.01 0.001]	[0.001 0.001 0.005]	[0.005 0.005 0.005]	[0.001 0.001]
<b>mid-noise</b>	[0.1 0.1 0.01]	[0.01 0.01 0.05]	[0.05 0.05 0.05]	[0.01 0.01]
<b>high-noise</b>	[1.0 1.0 0.01]	[0.01 0.01 0.1]	[0.5 0.5 0.5]	[0.1 0.1]

**Table 6.1.** Noise figures for the second set of synthetic experiments.

To investigate the effects of using multiple constraints for solving the Multi-Primitive Registration problem, we compared the convergence results of our solver using different factors combinations. A factor models a constraint between two matchables. In particular, we consider: all factor types (**all**), only homogeneous factors (**hom**: point-point, line-line, plane-plane) and only heterogeneous factors (**non-hom**: line-point, plane-point, plane-line).

For these experiments we track the error evolution for the different combinations of factors and we compare it with point only factors. To evaluate the performance of these approaches under varying noise conditions, we added to the original datasets noise sampled from  $\mathcal{N}_t(0, \Sigma_t)$ ,  $\mathcal{N}_R(0, \Sigma_R)$ ,  $\mathcal{N}_p(0, \Sigma_p)$  and  $\mathcal{N}_d(0, \Sigma_d)$  respectively for the translational and rotational part of the sensor pose and the point and direction of the matchable.

Then, we measured the convergence varying both the statistical parameters of the noise distributions, see Tab. 6.1, and the initial guess, computed from the optimum and the spanning tree. The graph is composed of 10000 poses and 18000 matchables. The optimization has been performed in the  $g^2o$  framework using the Levenberg-Marquardt solver. Both the implemented factors and scripts to reproduce the experimental evaluation are available online<sup>1</sup>.

In Fig. 6.4 we plot the evolution of the residual quadratic error, obtained by summing the components  $\mathbf{e}^{[n,m]}$  for each measurement  $\mathbf{z}^{[n,m]}$  (chi2), for 100 iterations. For more clarity, we normalized the chi2 according to its initial value. As evident from the plots, with all combinations of factors our solver converges to the optimum and, in general, displays better performance than the Multi-Point registration algorithm.

### 6.3.2 Real-world Experiments

With this set of experiments we want to evaluate the benefit of using global optimization for correcting the error in the camera pose estimation, that is usually accumulated while tracking. For this purpose, we considered the *fr2/desk* and *fr3/office* sequences of the TUM benchmark suite, since they contain a trajectory loop.

To conduct the experiments we first process a sequence with the tracker of Sec. 5.2. This allows to estimate the sensor pose for each RGB-D frame. At the same time, we use this information along with the extracted matchables to build the factor graph and detect closures, as explained in Sec. 6.2.2. Finally, we use our solver to optimize the graph.

Using the evaluation tool provided by the TUM benchmark we measure the absolute trajectory error (ATE) for both: the trajectory estimated with the tracker

<sup>1</sup>source: <https://github.com/istinj/g2o>



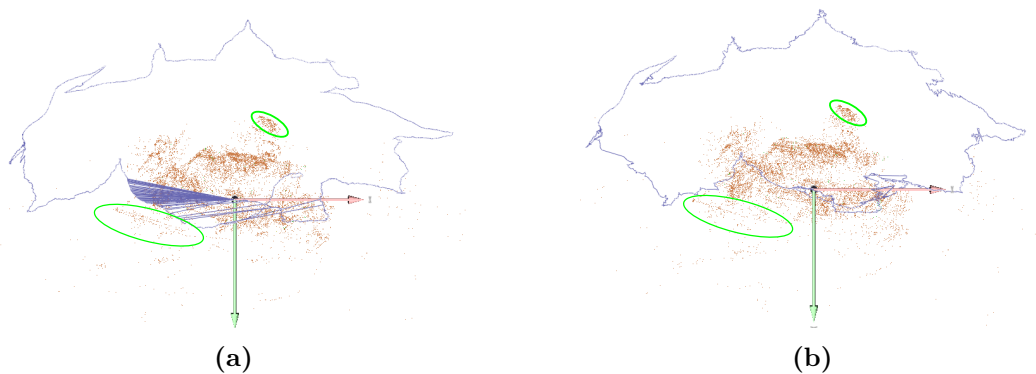
	<i>tracker-only</i>	<i>optimized</i>	DVO-SLAM
fr2/desk	0.238	0.144	<b>0.017</b>
fr3/office	0.296	0.197	<b>0.035</b>

**Table 6.2.** TUM - Absolute Trajectory Error [m].

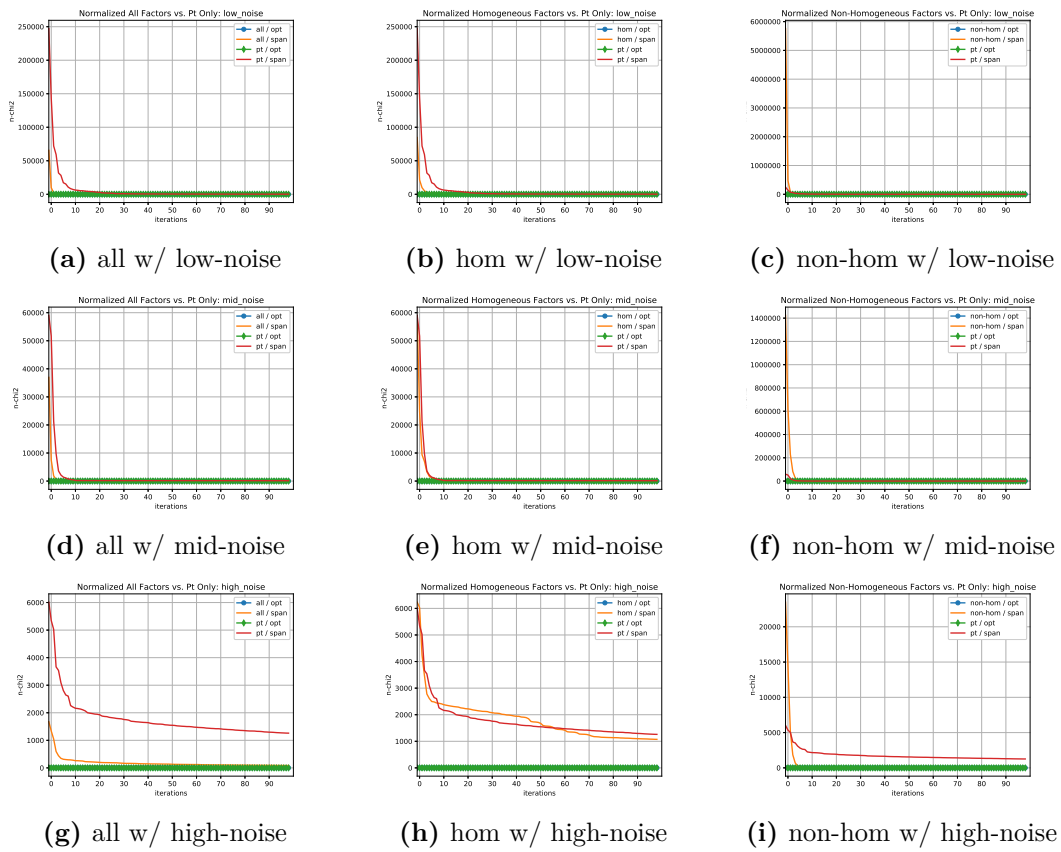
(*tracker-only*) and the one returned by the solver (*optimized*). Results are shown in (tab), where we report as reference the performance on the same sequences of DVO-SLAM [73].

The evaluation confirms that our global optimizer is capable of improving the estimation accuracy. This is visible from Fig. 6.3, where we have circled in green some of the corrections introduced by the method.

It is also evident that w.r.t to a state-of-the-art method there is a difference in performance of an order of magnitude. This may be credited to the simplistic processing pipeline used as front-end, since no local-map management has been implemented and the solver is called just once after all data has been processed. However, the results promote further investigation along this line.



**Figure 6.3.** Visual comparison of the two factor graphs rendered by the `g2o_viewer`: (a) before optimization (b) after optimization. Notice that we draw just the matchable point  $\mathbf{p}_m$ .



**Figure 6.4.** Synthetic experiments: normalized chi2 evolution. opt: starting from the optimal initial guess, span: initial guess from the spanning tree.

# Part III

## Semantic maps



## Chapter 7

# Taxonomy of a Semantic Mapping System

The Semantic Mapping problem raised particular interest of the robotics community in the last two decades and a deeper investigation is envisioned in the next years. In this chapter, we present a possible categorization of the several methods proposed in literature.

The underlying idea of this taxonomy is that a semantic mapping system involves the interplay of processing modules to solve different subproblems. By considering the subproblems which each method focuses on, we devised a system architecture that includes all of them and, at the same time, allows to compare them based on the particular techniques they adopt.

In general, a semantic mapping system should be made of three fundamental components (see Fig. 7.1): *perception*, *map construction* and *action*. In the remainder, we will delve deeper into each of them by describing its task and the proposed approaches to accomplish it. This survey is summarized in Tab. 7.1. The reader might notice that we refer to the methods reviewed in Sec. 2.3.

### 7.1 Perception

Mobile robots are usually equipped with exteroceptive sensors to sense the environment surrounding them. The *perception* module has the task of processing the measurements acquired with these sensors to extract geometric and/or semantic information. This is then inserted into the map to update the robot knowledge of the environment.

As discussed in Chapter 4, there are various ways to represent the geometric structure of the environment. In this context, we refer to this aspect as *reconstruction*. From a survey on relevant literature, we noticed that the most frequent representations are: 2D/3D occupancy grids and point clouds. Furthermore, it is also frequent to build a topological representation of the workspace that allows the robot to assess the connectivity between places and/or objects in the environment.

Next, with *recognition* we indicate the process of extracting patterns from sensor output representing objects/scenes categories and assign them the corresponding semantic label. As we have seen in Sec. 2.3.1, this problem has been extensively

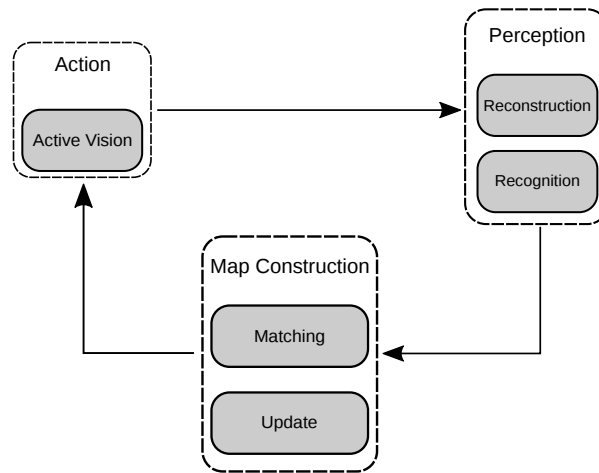


Figure 7.1. Semantic mapping system.

studied in Computer Vision and Robotics. For Semantic Mapping applications, the most common techniques are:

- **image classification:** assigning a semantic label to an image from a fixed set of categories.
- **object detection:** making a prediction not only of object categories but also of their spatial locations.
- **image segmentation:** finding groups of pixels that possess some "similarity" among each other.
- **scene labeling:** assigning a semantic label to each geometric feature of the scene.
- **3d object recognition:** given an input model and a 3d scene, finding the location of the input model inside the scene.

From this distinction, it is possible to discriminate methods that focus on the *perception* aspect according to the order in which they perform *reconstruction* and *recognition*. Possible approaches are:

- **geometric only:** consists in recovering only the geometric structure of the environment. This is the case, of maps that contain no semantic information.
- **pipelined:** consists in recovering a geometric model of the environment and, subsequently, using this model to extract semantic information.
- **parallel:** reconstruction and recognition are performed separately (and simultaneously) and then, their results are fused together.

## 7.2 Map Construction

The *map construction* module is in charge of incrementally updating the semantic map with new information coming from the *perception* module.

In order to renew the semantic map each time a sensor measurement is acquired, it is necessary to establish a correspondence between the percepts in the current sensor frame and the robot internal representation (*matching*).

After that, the incoming information is inserted in the map with an *update* procedure. Similar to the fusion strategy depicted in Fig. 4.1, this involves merging matched percepts, followed by adding unseen ones to the map. For semantic maps there are two possible ways of performing this step:

- **bayesian:** the update of a map element considers only past observations.
- **context-aware:** the update of an element considers also spatial relations with other elements.

Among the considered methods, the construction of the map is performed:

- **off-line:** the semantic map is built after all sensor measurements have been acquired by the robot during exploration.
- **frame-by-frame:** the map is built while the robot is exploring the environment and acquiring sensor measurements.

## 7.3 Action

The action module takes as input the updated map representation to plan a motion trajectory for the robot sensors. The objective of this plan is typically to improve the robot knowledge about the environment. In particular, possible objectives are:

- **reconstruction:** if the robot needs accurate 3D models of the objects present in the scene, as in the case of manipulation tasks.
- **recognition:** if the robot needs a precise categorization of the objects in the scene. As an example, this enables multi-view recognition approaches.

In general this problem is approached by first determining a space of candidate views for the sensor. Then, a metric is defined to perform an evaluation of the views and choose the best one. To find the *next best view*, possible approaches can be classified as:

- **search-based:** the trajectory is determined by computing a plan in the sensor configuration space.
- **optimization-based:** each new sensor pose is computed optimizing an objective function.

Approach	Perception	Reconstruction	Recognition	Map Construction	Update	Action	Active Vision
[173]	pipelined	2D occupancy grid + topological	object detection	frame-by-frame	bayesian		
[107]	pipelined	point cloud	scene labeling + 3D object recognition	offline			
[130]	pipelined	point cloud	scene labeling + 3D object recognition	offline		reconstruction	optimization
[41]	geometric only	voxel grid				reconstruction + recognition	optimization
[36]	pipelined	point cloud + topological	3D object recognition				
[153]	pipelined	point cloud	scene labeling + 3D object recognition	offline		reconstruction	search
[120]	geometric only	voxel grid				reconstruction	
[79]	pipelined	metric	scene labeling	offline		reconstruction	search
[155]	geometric only	voxel grid				reconstruction + recognition	search
[142]	parallel	point cloud	object detection + 3D object recognition				
[145]	parallel	voxel grid	semantic segmentation	frame-by-frame	bayesian + context-aware		
[121]	parallel	2D occupancy grid + topological	place categorization + object detection	frame-by-frame	bayesian + context-aware		
[161]	pipelined	point cloud	scene labeling	offline			
[80]	pipelined	point cloud	3D object recognition	frame-by-frame	bayesian	search	reconstruction + recognition
[60]	parallel	point cloud	semantic segmentation	frame-by-frame	bayesian + context-aware		
[162]	geometric only	voxel grid				reconstruction	search
[163]	parallel	point cloud	semantic segmentation	frame-by-frame	context-aware		
[172]	parallel	point cloud	object detection	frame-by-frame	bayesian	optimization	reconstruction + recognition
[98]	parallel	point cloud	semantic segmentation	frame-by-frame	bayesian + context-aware		

Table 7.1. Semantic Mapping Systems taxonomy.



## Chapter 8

# Generating a Semantic Map

To be fully operative in human environments, robots need particular information to be encoded in their representation of the workspace. As an example, if we want a service robot to accomplish the task “bring me a bottle of water”, it needs to have a notion of the items present in a domestic environment and their location. Likewise, if we want an autonomous car to drive in real traffic, it has to be equipped with some system to detect and recognize road signs as well as other cars and pedestrians.

As we have seen in Chapter 4, SLAM algorithms return a “metric” map. Augmenting this representation with information that may support more complex tasks is a well-known problem in Robotics and many solutions have been proposed (Sec. 2.3).

From Tab. 7.1 in Chapter 7, it is evident that existing methods mainly focused on a single aspect of the problem, namely: perception or action. On one hand, “perception” methods addressed the issue of attaching a semantic label to the geometric primitives of the metric representation, like points or voxels. Thus, the map is enriched with semantic information, but it is still inefficient to be used by the robot, because these primitives are too *low-level*.

On the other hand, “action” methods guarantee to obtain accurate 3D models of the elements that the robot has to interact with but they typically require a prior knowledge of the environment. This aspect severely limits their range of applicability, rendering them effective only if strong simplifying assumptions are satisfied.

In the remainder, we present a Semantic Mapping system that aims at bridging the gap between these two approaches by:

- Building an *object-based* map that can be used for complex task planning and execution.
- Exploiting efficient techniques for reconstruction and recognition that require no *a-priori* information and can be used under general conditions.

Our system follows the architecture defined in Fig. 7.1. In Sec. 8.1 we describe in detail each component. To study the feasibility of this approach a simulation environment has been used, the details of which, along with the conducted experiments, are presented in Sec. 8.2.

## 8.1 System Overview

As evident from recent years publications the range of mobile robotic applications is certainly wide. However, for our purpose, we need to deal with a setting whose specifications may be valid in general conditions. This allows to avoid addressing application-specific issues and to provide a general model of semantic mapping system.

Therefore we consider the setting of a service robot that has to operate in a household environment. Indoor scenarios facilitate the mapping task by allowing to exploit two basic assumptions:

- Most indoor settings have a flat floor, this enables the use of *out-of-the-box* 2D SLAM techniques to build an accurate metric map.
- RGB-D cameras are known to be better suited for indoor scenes because of the almost constant lighting conditions and they can be employed to obtain *dense* geometric and appearance information.

Conversely, if one wants to apply the same system in outdoor scenarios (like roads, harsh terrains or underground mines), for which robust mapping techniques exist [150, 97], it is sufficient to substitute some of the modules with the most suitable ones.

### 8.1.1 Perception

The *perception* module processes the image frames acquired with the RGB-D camera. Following the definitions in Sec. 7.1, our perception approach is parallel, that is, the depth and RGB images are processed separately.

The first is used to recover the robot pose and a point cloud expressed in global coordinates (*reconstruction*). While, from the latter we infer the semantic class of each pixel (*recognition*).

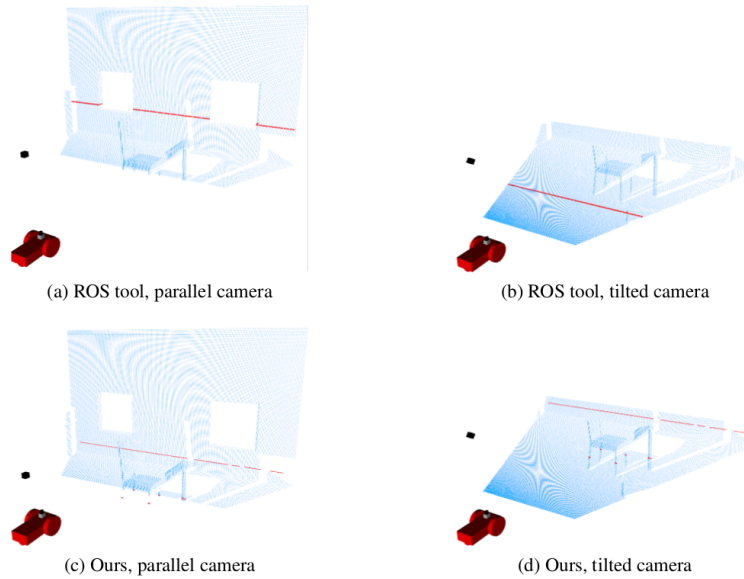
By combining this information, we extract a set of objects, see Sec. 8.1.2. Subsequently, this will be passed to the *map construction* module and used to update the map.

### Reconstruction

To estimate the robot pose, we follow the procedure presented in [101]. This consists in extracting a *virtual* laser scan from the depth image (Fig. 8.1) and feed it to a robust 2D SLAM algorithm [86].

Having estimated the pose of the robot w.r.t. a global reference frame, we are able to exploit this information for building 3D models of the scene elements. This is eventually done by fusing together partial views of each object in its corresponding model. In this work, we model objects surface with 3D point clouds and we make use of the Point Cloud Library (PCL) [129] to process them.

As a first step, we need to convert the depth image  $D$  into a point cloud. To this end, it is necessary to apply an unprojection function  $\pi^{-1}$  that depends on the



**Figure 8.1.** Comparison of the extraction procedure with the ROS *depthimage\_to\_laserscan* tool and our method in [101]. The RGB-D camera is represented by the black box on top of the robot, the virtual laser is represented by the 3D model of an Hokuyo sensor.

intrinsic camera parameters. A common solution is to assume a pinhole camera model, whose calibration matrix [58] is:

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad (8.1)$$

with  $f$  focal length and  $(p_x, p_y)^\top$  coordinates of the principal point. For a depth camera, the unprojection function will be:

$${}^c\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \pi^{-1}(u, v, d) = \mathbf{K}^{-1}d \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} (u - p_x)d/f \\ (v - p_y)d/f \\ d \end{pmatrix} \quad (8.2)$$

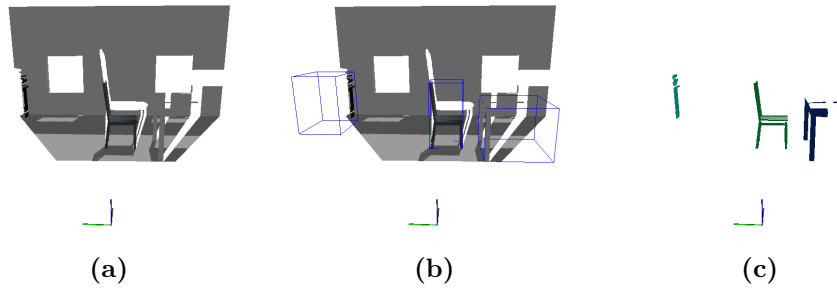
with  $d = D(u, v)$  the depth value at the pixel  $(u, v)$ . The set of 3D points obtained in this way is expressed in the camera reference frame. Considering the camera pose w.r.t the map  ${}^mT_c$ , it is possible to transform these points in the global reference frame with the simple operation:

$${}^g\mathbf{x} = {}^gT_c {}^c\mathbf{x}. \quad (8.3)$$

Next, we will see how to identify to which object each point belongs.

## Recognition

The RGB image is processed to detect and localize objects in the environment. To this end, various Computer Vision techniques are available. In a single frame



**Figure 8.2.** Semantic segmentation procedure: (a) input point cloud (b) models bounding boxes superimposed to the cloud (c) points belonging to the detected objects (each one is colored depending on the object type).

different objects may be captured by the camera, thus, simple *image classification* is not sufficient. In this case, *object detection* is better suited. However, the typical output of an object detector is a set of 2D bounding boxes, each labeled with a semantic class. This is a rather imprecise information for determining the exact 3D position of the objects in the scene.

For this reason, we decided to apply a *semantic segmentation* technique. In this way, we obtain a *dense* classification, that is, each pixel of the image gets assigned a label. This is a convenient information because it allows to group together pixels marked with the same label and from that we can obtain the 3D points that belong to each partial view of the objects.

Given an RGB image  $I$  and a set of semantic labels  $\mathcal{L} = \{l_1 \dots l_N\}$ , *semantic segmentation* consists in computing an assignment  $L(\mathbf{x})$  of labels  $l_i$  to each image pixel  $\mathbf{x} = I(u, v)$ . In a simulated environment it is possible to execute this task by performing geometrical computations.

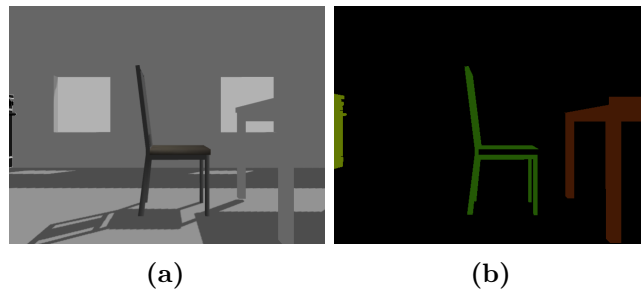
The camera field-of-view can be defined through: aspect ratio, horizontal angle-of-view and far and near clipping planes. Knowing the sensor pose, this allows to infer which objects the sensor is framing with a common Computer Graphics technique called view frustum culling [2]. The result is a list of objects with the corresponding 3D bounding boxes.

After that, it is necessary to evaluate which elements in the view frustum are visible by the robot, and which are hidden. This is typically done with the so-called Z-buffer algorithm [50]. We can assume that the depth image is indeed a Z-buffer, thus, we use this information for occlusion check.

Given the depth point cloud and the models bounding boxes, for each point we compute to which model it belongs (see Fig. 8.2). Then, we mark the corresponding image pixel with the model label. The result of this procedure is shown in Fig. 8.3.

### 8.1.2 Map Construction

A common approach to build an *object-based* map is to build a database of CAD models and perform 3D Object Recognition to determine the pose and the type of the various objects that constitute the scene [132, 21, 28]. Then, this kind of representation can be used for tasks like human-robot interaction or manipulation. However, this method has two shortcomings:



**Figure 8.3.** Semantic segmentation result: (a) input (b) output. Pixels that do not belong to an object are in black.

- 3D Object Recognition is a computationally expensive process and it fails when the points sampled from an object surface are not dense enough, *e.g.* for small objects or distant from the sensor.
- The requirement of building a models database beforehand limits the generality of the method, especially in the context of human environments which are mostly unpredictable.

The alternative is to adopt an incremental approach to build the object models. This is similar to the process described in Chapter 4, with the difference that, instead of low-level primitives, in this case we consider a map composed of objects. In the remainder, we will give a formal definition of *object-based* map and present the two main steps to build it.

## Representation

In our system an *object-based* semantic map is a collection of elements:

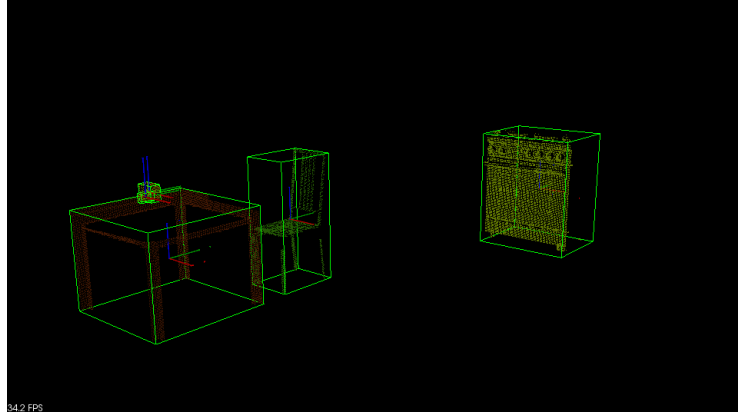
$$\mathcal{SM} = \{\mathbf{E}_1, \dots, \mathbf{E}_N\}, \quad (8.4)$$

where each element  $\mathbf{E}_i$  is characterized by both geometric and semantic information. To fully qualify an element geometry we make use of:

- **position:**  $\mathbf{p} \in \mathbb{R}^3$  expressed in the global reference system.
- **size:**  $\mathbf{S} = \langle \mathbf{L}, \mathbf{U} \rangle \in \mathbb{R}^3$  represented by a pair of 3D vectors that define the element bounding box.
- **model:**  $\mathbf{M} = \{\mathbf{x}_{1:N}\}$  a point cloud representing the element surface.

Moreover, each element of the map can be described by the following semantic properties:

- **type:**  $t \in \mathcal{L}$  the element category. It is selected by the *recognition* module among the possible semantic labels  $\mathcal{L} = \{l_1 \dots l_M\}$ .
- **properties:**  $\mathbf{P} = \{p_1 \dots p_K\}$  other semantic information related to the element, *e.g.* relationships with other elements, physical properties, functionalities.



**Figure 8.4.** Representation of the *object-based* map. Each element has a: type (color), position (axes), size (bounding box) and model (point cloud).

The *map construction* module has the task of building these elements. It does so by continuously integrating into the map the information coming from the *perception* module, see Fig. 8.4 for an example.

To this end, first it is necessary to establish a correspondence between the currently observed objects and their representation in the map. Then it is possible to: (1) update the representation of objects that have already been observed and (2) add the unobserved ones.

### Matching

We refer to the list of elements returned by the *perception* module as a local map  $\mathcal{SM}^L$ . Where the term *local* indicates that it represents a limited portion of the environment. Consequently, we refer to the representation built so far as the global map  $\mathcal{SM}^G$ .

Matching consists in finding a set of correspondences  $\mathcal{C} = \{\mathbf{c}_1^{[i,j]}, \dots, \mathbf{c}_K^{[i,j]}\}$  between elements in the two maps, with  $\mathbf{c}_k^{[i,j]} = \langle \mathbf{E}_i^L, \mathbf{E}_j^G \rangle$ . Such a correspondence can be assessed through a similarity measure  $d(\mathbf{E}_i^L, \mathbf{E}_j^G)$ . Different choices of  $d$  are possible. In this work, we found that the most distinctive attributes of an element are the type and the position, *i.e.* they are informative enough to uniquely identify it. To measure their similarity, we first check that two objects belong to the same semantic class. If this is the case, we compute the distance between their position, otherwise we set it to a big arbitrary number.

Therefore, given the global map  $\mathcal{SM}^G$  and the local map  $\mathcal{SM}^L$ , for each element  $\mathbf{E}_i^L$ , we want to find the element  $\mathbf{E}_j^{G*}$  that minimizes  $d$ :

$$\mathbf{E}_j^{G*} = \underset{\mathbf{E}_j^G}{\operatorname{argmin}} d(\mathbf{E}_i^L, \mathbf{E}_j^G). \quad (8.5)$$

In our system we find the set of correspondences  $\mathcal{C}$  through a Nearest Neighbor Search.

```

Require:  $\mathcal{SM}$ : map,  $T_r$ : robot pose
1: exit = false;
2: while !exit do
3:    $\mathbf{E}_n \leftarrow \text{findNearestObject}(\mathcal{SM}, T_r)$ ;
4:   if ! $\mathbf{E}_n$  then
5:     exit = true;
6:     continue;
7:   end if
8:   reconstructed = false;
9:   while !reconstructed do
10:     $\mathcal{SM} \leftarrow \text{receiveMap}()$ ;
11:     $T_r \leftarrow \text{receiveRobotPose}()$ ;
12:     $V \leftarrow \text{generateCandidateViews}(\mathbf{E}_n)$ ;
13:    reached = false;
14:    while ! $V.\text{empty}()$  & !reached do
15:       $[v^*, s_{unn}] \leftarrow \text{computeNBV}(V)$ ;
16:      if  $s_{unn} < \alpha_{new}$  then
17:        reconstructed = true;
18:        continue;
19:      end if
20:      reached  $\leftarrow \text{goToPose}(v^*)$ ;
21:       $V.\text{pop}()$ ;
22:    end while
23:  end while
24: end while

```

**Algorithm 5:** Active vision strategy

## Update

After the matching procedure, we iterate through the set of correspondences  $\mathcal{C}$  to update the global map. In principle for each object both the geometric and the semantic attributes should be updated.

Regarding the semantic part, typical CNNs return a confidence score about their predictions. This should be averaged to the value obtained from past observations. Since in our system this module is simulated, we are not considering this aspect and we update only the geometric information.

To do so, for each correspondence  $\mathbf{c}_k^{[i,j]}$ , we first merge the models  $\mathbf{M}^G$  and  $\mathbf{M}^L$  of the two objects. This is done by adding the two point clouds and then downsampling the result with a VoxelGrid filter (using the PCL implementation). After that we compute: (a) the new object position  $\mathbf{p}^G$ , by extracting the cloud centroid and (b) the new bounding box  $\mathbf{S}^G$ , by computing the minimum and maximum values along the  $x$ ,  $y$  and  $z$  dimensions of the point cloud  $\mathbf{M}^G$ . Finally, the objects in the local map that have not been associated are simply inserted in the global map.

### 8.1.3 Action

So far, we referred to incremental map building as the process of integrating sensor measurements acquired along a motion trajectory but nothing has been said about the nature of the trajectory. Where possible, a practical solution is to have a human teleoperating the robot. In other cases, *e.g.* for planetary rovers or robots designed to map dangerous areas, automatic exploration techniques are required.

For wheeled robots the terrain surface on which they navigate can be represented with Occupancy Grids or Digital Elevation Maps (DEMs). Common exploration approaches adopt frontier-based strategies [168, 136]. On a grid, frontiers are sets of adjacent cells that have at least a neighboring cell whose value is unknown. Once frontiers are detected on the map, a score is assigned to each of them depending on some criteria (*e.g.* information gain, distance and bearing w.r.t the robot, traversability) and the robot is directed towards the top scoring one. Another strategy to detect frontiers is based on randomized search techniques that sample the robot configuration space and find traversable paths to expand the mapped area [85, 42].

These methods guarantee a complete coverage of the area to map. However, for our purpose, it is not enough. Borrowing the terminology from [3], navigation tasks require “perception in the large”. This is because the main objective is to conduct the robot towards a goal destination while taking care of avoiding obstacles. In this sense, it is sufficient to detect the area the latters occupy and overestimating it does not cause problems. On the other side, manipulation tasks require “perception in the small”. In this case, it is necessary to accurately recover the 3D geometry of the objects in order to plan a motion trajectory for the end-effector. To this end, we integrate an *active vision* strategy in our system.

Our approach derives from the NBV framework and we make use of the Octomap library [64] to obtain a volumetric reconstruction of an object. This allows to perform *view evaluation* through *ray-casting* by computing a score based on the number of unknown traversed cells. In the remainder we provide details for each step of the procedure. A pseudo-code sketch is shown in Algorithm 5.

#### Active Vision

To start building the map we place the robot in its initial position and let it acquire measurements of the scene. They are processed by the *perception* and *map construction* modules as described in the previous sections. In this way it is possible to trigger the active vision strategy by providing it with the map and the robot pose.

As defined in Sec. 8.1.2, the map  $\mathcal{SM}$  is composed of a list of objects. To actively reconstruct them, we consider the closest one, according to the robot pose  $T_r$ . To this end, we iterate through the list of objects. For each of them, if it has not been yet processed, we compute the *euclidean distance* from the robot and we set the nearest  $\mathbf{E}_n$  as the one to be reconstructed next.

A crucial requirement to compute the NBV is a volumetric reconstruction of the object. The Octomap library allows to build it from a 3D point cloud. This is done by first building an *octree* whose dimensions are defined according to the object bounding box. After that, the *octree* will contain only unknown voxels. Then,

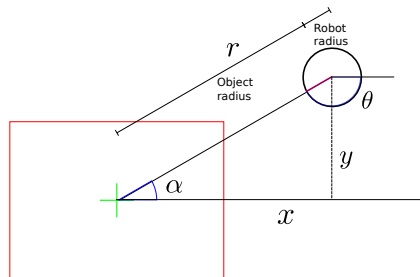


**Require:**  $N$ : number of poses,  $r$ : radius

**Ensure:** Set of candidate poses  $V$

- 1: **for**  $i = 0 : N - 1$  **do**
- 2:    $\alpha = i \cdot 2\pi/N$
- 3:    $x = r \cdot \cos(\alpha)$
- 4:    $y = r \cdot \sin(\alpha)$
- 5:    $\theta = \text{atan2}(-y, -x)$
- 6:    $V \leftarrow [\mathbf{E}_n \cdot x + x, \mathbf{E}_n \cdot y + y, \theta]$
- 7: **end for**

**Algorithm 6:** Candidate views computation



**Figure 8.5.** Candidate pose computation.

a scan is generated for each point to mark the free and occupied voxels.

To generate the candidate views we define a set of 2D poses  $V$  on a circle around the object to reconstruct, each of which points towards its centroid. To do so, we follow the procedure in Algorithm 6, where we usually set  $r$  as the sum of the object and the robot radius. As shown in Fig. 8.5, the object radius is chosen as the biggest horizontal size of its bounding box.

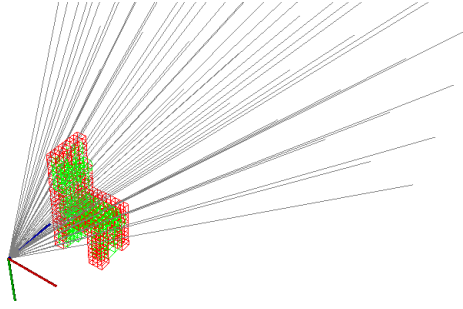
After that, we evaluate each pose  $v \in V$  by generating a *virtual* view. This is done by casting a set of rays and counting how many unknown voxels are traversed. The rationale behind this approach is that we want to maximize the information gain of each partial view.

A ray is a 3D line that can be defined in different ways. Since we are using the `computeRay()` function from the Octomap library, a ray has to be parameterized through an *origin* and an *endpoint*. To compute the *virtual* view we let each ray start from the camera origin and pass through a pixel of the sensor. Recalling that each 2D point  $(u, v)$  on the image plane backprojects to a 3D line [58], we compute a ray endpoint with the following formula:

$$\mathbf{x}_{end} = \underbrace{\mathbf{K}^{-1}}_{\substack{\text{Inverse} \\ \text{camera} \\ \text{matrix}}} \cdot (u, v, 1)^\top. \quad (8.6)$$

Then, we set the length of this ray to  $l_{max}$  by normalizing  $\mathbf{x}_{end}$  and multiplying it by  $l_{max}$ . Finally, we use the robot pose to transform it into global coordinates. We repeat this computation for each pixel of the image, the result is shown in Fig. 8.6.

For each view we count how many unknown cells are hit by the rays and store it



**Figure 8.6.** Ray-casting procedure. The axes show the candidate view, gray lines represent the rays. The volumetric reconstruction is shown with wireframe cubes: occupied (red), green (free).

in  $s_{unn}$ . Consequently, the score for each view is computed as:

$$s(v) = s_{obs} \cdot s_{cost} \cdot s_{unn}, \quad (8.7)$$

where  $s_{obs}$  is 1 when the candidate pose is reachable, 0 otherwise and  $s_{cost}$  is the inverse of the path cost returned by the planner. The best scoring pose  $v^*$  is chosen as next destination for the robot and while it is reached we acquire new measurements. These update the map, including the object model, and are then integrated in its volumetric reconstruction.

The computation of candidate views and their evaluation is repeated until no new relevant information can be added to the object model. That is,  $s_{unn}$  is below a threshold  $\alpha_{new}$ . Consequently, the object is inserted in the *processed* list and we search again for the nearest one. The algorithm terminates when there are no more objects to reconstruct.

## 8.2 Experimental Evaluation

The system described in this chapter is made of a combination of different *hardware* and *software* components and implementing it would require a considerable amount of human and economic resources. A common pattern in robotic applications is to rely on a *simulation engine* to study the feasibility of a system. The benefit is twofold: (1) it reduces the time needed to setup the experimental setting and (2) it abstracts from real-world complexity allowing to focus on the core processing aspects.

In this work we used the Gazebo simulator<sup>1</sup>, which provides a basic physics engine and several customization options. The first step to setup the environment for experiments has been to extend the models database. Since we want our robot to operate in a household setting, we retrieved from the Google 3D Warehouse<sup>2</sup> the CAD models of different pieces of furniture and imported them into Gazebo.

<sup>1</sup>wiki: <http://gazebosim.org/>

<sup>2</sup>web: <https://3dwarehouse.sketchup.com/?hl=it>

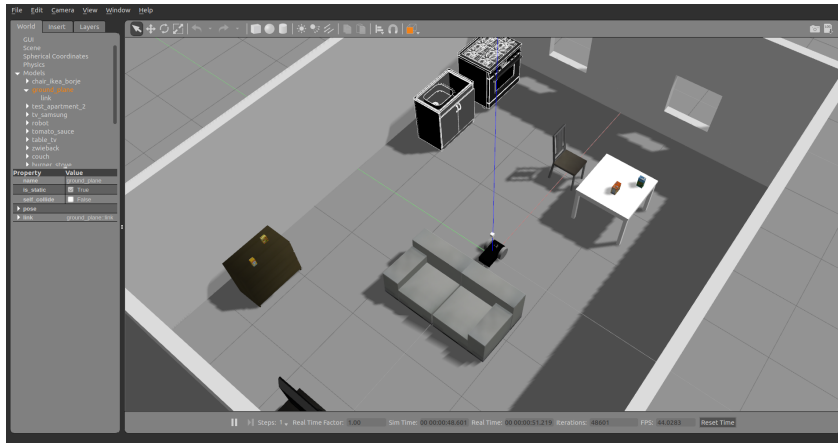


Figure 8.7. Simulation environment for experiments.

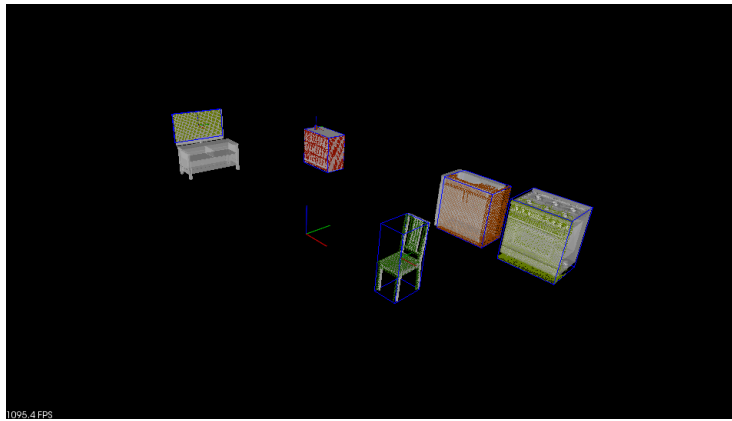


Figure 8.8. Example evaluation of the reconstructed map. Grey meshes are obtained from Gazebo models. Reconstructed objects are represented through: reference frame (position), blue box (size), colored point cloud (3D model).

Subsequently, we designed our robotic platform. For this setting we used a *differential-drive* wheeled robot equipped with an RGB-D camera. The Robot Operating System (ROS)<sup>3</sup> allows to represent a robot model through the Unified Robot Description Format (URDF) in terms of links, joints, actuators and sensors. This format is also supported in Gazebo and acts as an interface between the two frameworks. On one side, Gazebo provides the physics to simulate the behavior of sensors and actuators while, on the other side, ROS allows to process the gathered data and send commands to the platform.

The simulation environment and the robotic platform to run the experiments are shown in Fig. 8.7. The models used in simulation along with the configuration scripts to execute the experiments are available online<sup>4</sup>.

<sup>3</sup>web: <http://www.ros.org/>

<sup>4</sup>source: [https://github.com/schizzz8/lucrezio\\_simulation\\_environments](https://github.com/schizzz8/lucrezio_simulation_environments)

	Position Error [m]	Size Error [m <sup>3</sup> ]
burner_stove	0.00566078	0.00936389
cabinet_ikea_malm_big	0.00231613	0.00173253
chair_ikea_borje	5.59976e-04	5.21541e-04
couch	0.000113901	0.000226796
milk	0.0605588	0.000696493
salt	0.0247678	0.000261408
sink	0.0118303	0.0163019
table_ikea_bjursta	1.00426e-03	2.18749e-03
table_tv	0.0987725	0.096771
tomato_sauce	0.0189905	0.000209789
tv_samsung	1.31447e-04	4.17084e-04
zwieback	0.0139543	0.00031704

**Table 8.1.** Position and size errors for the reconstructed objects.

### 8.2.1 Constructing the Map

As a first experiment, we test the *map construction* module. For this purpose, we run the ROS node<sup>5</sup> that implements this functionality and we move the robot within the environment by tele-operating it.

On a mobile Intel Core i7-4710HQ with 2.50 GHz the execution time of the semantic segmentation procedure is  $\sim 0.2s$ . Therefore, we do not process every frame from the RGB-D camera, instead, the map construction procedure is triggered every second. As in [60], this is not a limitation, since in most typical scenarios the movement between consecutive frames is small and the recognition process does not need to be real-time capable.

For reconstruction and recognition techniques different evaluation metrics exist. For SLAM algorithms one typically measures the Absolute Trajectory Error (ATE), while, classification can be evaluated through “precision and recall” curves or F1 score. However, one of the advantages of using a simulation environment is that of having a *ground-truth* for the map. This is contained in the configuration files to launch the simulation and can be used to measure the accuracy of the reconstructed objects (Fig. 8.8).

In particular, we measure the position and size error between the reconstructed and the ground-truth objects. The position error is the norm of the difference between the two position vectors, while the size error is computed by subtracting the volumes of the two objects measured from the corresponding bounding boxes. Tab. 8.1 reports these errors for the tele-operated experiment.

The reader might notice that all errors are very close zero. This is because the algorithm works in ideal conditions. However, in the remainder we will make some considerations about more realistic scenarios.

For the reconstruction aspect, it is likely that the pose estimate may drift. Nevertheless, especially in the 2D case, graph-based approaches may still be able to recover a globally consistent map (see Sec. 2.2.1). Therefore, in this case a viable solution would be to keep in memory the partial views of the object to reconstruct (by simply storing the point cloud and a pose identifier). Then, when a loop closure occurs, the objects can be rendered from the stored views and the corrected poses.

<sup>5</sup>source: [https://github.com/schizzz8/lucrezio\\_semantic\\_mapper](https://github.com/schizzz8/lucrezio_semantic_mapper)

Moreover, there are more sophisticated models for the fusion problem. One of that is the TSDF, Sec. 4.3 that is more robust to noise than a point cloud and can provide smoother models.

For the recognition aspect, it is known that no existing method is able to guarantee 100% accuracy. This typically results in false positives and false negatives. A possible strategy to tackle these issues would be to insert an object in the map only if it has been detected for a certain number of consecutive frames. This allows to filter “outliers” and avoids populating the map with inconsistent information. Moreover, recent CNNs return a confidence value about their prediction. This information can be considered as a further tool for preserving consistency.

### 8.2.2 Exploring the Environment

The comparison between two semantic mapping systems is hardly possible because: (a) there is not yet a common agreement on standard evaluation metrics and benchmarks and (b) usually no open-source implementations are available.

Therefore, we limit ourselves to show that indeed the proposed representation is usable by the robot, in this case to plan a trajectory for its sensor, and it can be built from scratch without a pre-built models database.

Furthermore, to evaluate the performance of the *active vision* strategy presented in Sec. 8.1.3, we compare it with a frontier-based one. Both strategies are implemented as ROS nodes and available online<sup>6</sup>. For the robot motion planning and execution, we rely on the `move_base` planner implemented in ROS<sup>7</sup>.

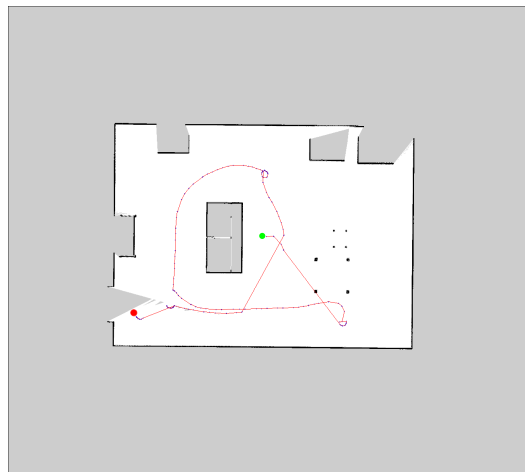
For both runs we report the computed motion trajectories in Fig. 8.9 and the position and size errors for the reconstructed objects in Tab. 8.2. The frontier-based strategy just aims at maximizing the area perceived by the sensor. This results in a faster execution time (72.046s) and a shorter trajectory. However, as expected, the obtained models are only partially reconstructed. On the other hand, at a cost of an higher execution time (423.118s), the active vision method allows to obtain more accurate models.

As a final remark. For the active vision strategy, it is clear that searching only in 2D space is a strong limitation. By mounting the sensor on a manipulator arm, it is possible to exploit more degrees of freedom and provide more complete reconstructions. However, the main problem with the augmentation of the search space is of course an increase in computational complexity. For this reason, this seems fertile ground for a learning-based approach and interesting investigations are appearing in this context [69].

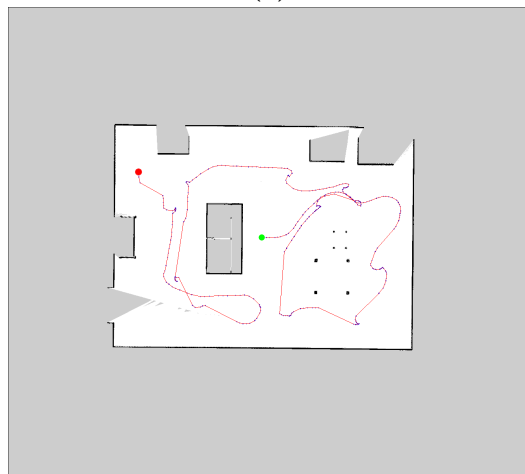
---

<sup>6</sup>source: [https://github.com/schizzz8/lucrezio\\_semantic\\_explorer](https://github.com/schizzz8/lucrezio_semantic_explorer)

<sup>7</sup>wiki: [http://wiki.ros.org/move\\_base](http://wiki.ros.org/move_base)



(a)



(b)

**Figure 8.9.** Motion trajectories for both exploration strategies: (a) frontier-based, (b) object-based. Green dot: starting point. Red dot: end point.

	<i>frontier-based</i>		<i>object-based</i>	
	<b>Position Error [m]</b>	<b>Size Error [m<sup>3</sup>]</b>	<b>Position Error [m]</b>	<b>Size Error [m<sup>3</sup>]</b>
burner_stove	0.0189602	0.0271541	0.00162562	0.00268817
cabinet_ikea_malm_big	0.0861186	0.116559	0.0027411	0.00204489
chair_ikea_borje	0.000802786	0.000677571	3.9217e-04	3.62098e-04
couch	0.00352228	0.0105351	0.00467137	0.0141263
milk	0.00829845	0.00015917	0.019375	0.000233552
salt	0.0476797	0.000395995	0.0151248	0.000198576
sink	0.102379	0.108796	0.00660678	0.0184774
table_ikea_bjursta	0.000147986	0.000270009	1.11544e-03	7.98702e-03
table_tv	0.00142145	0.00171903	0.0217376	0.0213282
tomato_sauce	0.0477151	0.000356746	0.00860188	9.31322e-05
tv_samsung	0.0422653	0.122738	3.04694e-04	6.14524e-04
zwieback	0.0106996	0.000259349	0.0166117	0.000390024

**Table 8.2.** Comparison of position and size errors for the two exploration strategies.





**Part IV**  
**Conclusions**



## Chapter 9

# Final Discussion

In many social and professional contexts there is the request by humans of being assisted by technology. Examples of this need are: (1) autonomous cars that should replace humans to provide more safe and efficient transportation (2) service robots intended to facilitate domestic daily tasks. At the time of writing, the general feeling is that these robotic applications are close to leave research laboratories to reach a broader public. However, this step poses a big challenge.

In order to collaborate with or replace humans in their activities, robots need a digital representation of the environment. This can be used for different operations, like: localization in the environment, recognition of objects or planning actions to complete a task. For us humans, these operations are fundamental for survival and are learned in the early ages of our life. Therefore, they seem to us pretty natural.

On the contrary, reproducing these behaviors on a computer has shown to be challenging both in terms of: (a) defining and building mathematical models that are able to describe man-made environments at a level of abstraction that captures their complexity (b) implementing strategies to process the information encoded in these models and produce the sequence of instructions that allow the machine to execute the desired actions.

These challenges have been accepted by scientific researchers and constitute the main fields of investigation for Mobile Robotics and Artificial Intelligence. In this context, the research work presented in this thesis addresses the general problem of robot mapping and localization and aims at contributing novel and relevant algorithms in the field.

### 9.1 Metric Maps

The main technical contributions start presenting a taxonomy of metric map representations in Chapter 4. In Chapter 5 we propose a unified representation for different types of primitives such as points, lines and planes. This approach allows to devise a registration algorithm for aligning hybrid scenes in a unifying formulation. By allowing different hybrid associations to contribute to the objective function, the proposed algorithm can potentially use more varied sources of information than any of the specific ICP variants. We evaluate the method on standard benchmarking datasets and show that it performs on par with current state-of-the-art registra-

tion methods. Similarly, we exploit the proposed representation to derive a global optimization algorithm which aims at aligning geometric primitives observed in different scans into a consistent map. The proposed global optimization scheme is evaluated as a SLAM back-end in a mapping system. The experiments confirm that the method is capable of improving the estimation accuracy. However results lag behind the state of the art due to the simplified front-end.

## 9.2 Semantic Maps

The second part of the thesis is focused on the usability aspect of environment representations. Chapters 7 and 8 present the contributions aimed at adding semantics to the map. This allows to build object-centered environment models that can support more complex robot operations. To this end, we first highlight which are the three fundamental aspects to consider in addressing this issue. Based on this aspects, we both propose a taxonomy of existing methods and a general architecture of a Semantic Mapping system that is capable of building object-based maps with no prior information about the environment. This is an advancement in the state-of-the-art since no other method considered jointly these aspects and still it is not available a system that can work under general conditions. Due to the complexity of actually implementing this system, its feasibility has been demonstrated in simulation.

## 9.3 Future Directions

Beyond these results, the proposed solutions indicate research lines to further implement the deployment of robots in man-made environments. In particular it may be relevant investigating about the extension of the representation introduced in Chapter 5 to model also objects, *e.g.* as a composition of simpler primitives. Likewise, based on the positive results obtained in simulation, the real implementation of the mapping system proposed in Chapter 8 can be of interest for the community.

# Bibliography

- [1] AIGER, D., MITRA, N. J., AND COHEN-OR, D. 4-points congruent sets for robust pairwise surface registration. In *ACM Transactions on Graphics (TOG)*, vol. 27, p. 85. ACM (2008).
- [2] ASSARSSON, U. AND MOLLER, T. Optimized view frustum culling algorithms for bounding boxes. *Journal of graphics tools*, **5** (2000), 9.
- [3] BAJCSY, R., ALOIMONOS, Y., AND TSOTSOS, J. K. Revisiting active perception. *Autonomous Robots*, **42** (2018), 177.
- [4] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *European conference on computer vision*, pp. 404–417. Springer (2006).
- [5] BELLEKENS, B., SPRUYT, V., BERKVEN, R., PENNE, R., AND WEYN, M. A benchmark survey of rigid 3d point cloud registration algorithm. *Int. J. Adv. Intell. Syst.*, **8** (2015), 118.
- [6] BERGER, M., TAGLIASACCHI, A., SEVERSKY, L., ALLIEZ, P., LEVINE, J., SHARF, A., AND SILVA, C. State of the art in surface reconstruction from point clouds. In *EUROGRAPHICS star reports*, vol. 1, pp. 161–185 (2014).
- [7] BESL, P. AND MCKAY, N. D. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **14** (1992), 239.
- [8] BESL, P. J. AND MCKAY, N. D. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, pp. 586–607. International Society for Optics and Photonics (1992).
- [9] BREIMAN, L. Random forests. *Machine learning*, **45** (2001), 5.
- [10] BRICE, C. R. AND FENNEMA, C. L. Scene analysis using regions. *Artificial intelligence*, **1** (1970), 205.
- [11] BRUCKER, M., DURNER, M., AMBRUŞ, R., MÁRTON, Z. C., WENDT, A., JENSFELT, P., ARRAS, K. O., AND TRIEBEL, R. Semantic labeling of indoor environments from 3d rgb maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1871–1878. IEEE (2018).

- [12] CALONDER, M., LEPETIT, V., STRECHA, C., AND FUA, P. Brief: Binary robust independent elementary features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 778–792 (2010).
- [13] CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (1986), 679.
- [14] CASTELLANOS, J. A., MONTIEL, J., NEIRA, J., AND TARDÓS, J. D. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on robotics and Automation*, **15** (1999), 948.
- [15] CHAN, T. F. AND VESE, L. A. Active contours without edges. *IEEE Transactions on image processing*, **10** (2001), 266.
- [16] CHEN, C.-S., HUNG, Y.-P., AND CHENG, J.-B. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21** (1999), 1229.
- [17] CHEN, J., BAUTEMBACH, D., AND IZADI, S. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)*, **32** (2013), 113.
- [18] CHEN, Y. AND MEDIONI, G. Object modelling by registration of multiple range images. *Image and vision computing*, **10** (1992), 145.
- [19] CHENG, Z.-Q., CHEN, Y., MARTIN, R. R., LAI, Y.-K., AND WANG, A. Supermatching: Feature matching using supersymmetric geometric constraints. *IEEE Transactions on Visualization and Computer graphics*, **19** (2013), 1885.
- [20] CHOI, C., TREVOR, A. J., AND CHRISTENSEN, H. I. Rgb-d edge detection and edge-based registration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1568–1575. IEEE (2013).
- [21] CIVERA, J., GÁLVEZ-LÓPEZ, D., RIAZUELO, L., TARDÓS, J. D., AND MONTIEL, J. Towards semantic slam using a monocular camera. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1277–1284. IEEE (2011).
- [22] COLE, D. M. AND NEWMAN, P. M. Using laser range data for 3d slam in outdoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1556–1563. IEEE (2006).
- [23] COMANICIU, D. AND MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **24** (2002), 603.
- [24] CORTES, C. AND VAPNIK, V. Support-vector networks. *Machine learning*, **20** (1995), 273.

- [25] CROCCO, M., RUBINO, C., AND DEL BUE, A. Structure from motion with objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4141–4149 (2016).
- [26] CURLESS, B. AND LEVOY, M. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312. ACM (1996).
- [27] DALAL, N. AND TRIGGS, B. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893. IEEE (2005).
- [28] DAME, A., PRISACARIU, V. A., REN, C. Y., AND REID, I. Dense reconstruction using 3d object shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1288–1295 (2013).
- [29] DAVIS, T. A. *Direct methods for sparse linear systems*. SIAM (2006).
- [30] DELLA CORTE, B., BOGOSLAVSKYI, I., STACHNISS, C., AND GRISETTI, G. A general framework for flexible multi-cue photometric point cloud registration. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, (2018).
- [31] DELLAERT, F. AND KAESSE, M. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, **25** (2006), 1181.
- [32] DESOLNEUX, A., MOISAN, L., AND MOREL, J.-M. Meaningful alignments. *International Journal of Computer Vision*, **40** (2000), 7.
- [33] DO CARMO, M. P. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications (2016).
- [34] DUCKETT, T., MARSLAND, S., AND SHAPIRO, J. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, **12** (2002), 287.
- [35] EADE, E. AND DRUMMOND, T. Edge landmarks in monocular slam. *Journal on Image and Vision Computing (IVC)*, **27** (2009), 588.
- [36] EIDENBERGER, R. AND SCHARINGER, J. Active perception and scene modeling by planning with probabilistic 6d object poses. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1036–1043. IEEE (2010).
- [37] ERHAN, D., SZEGEDY, C., TOSHEV, A., AND ANGUELOV, D. Scalable object detection using deep neural networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2154 (2014).
- [38] FELZENSZWALB, P. F. AND HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *Intl. Journal of Computer Vision (IJCV)*, **59** (2004), 167.

- [39] FENG, C., TAGUCHI, Y., AND KAMAT, V. R. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6218–6225. IEEE (2014).
- [40] FISCHLER, M. A. AND BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24** (1981), 381.
- [41] FOISSOTTE, T., STASSE, O., ESCANDE, A., WIEBER, P.-B., AND KHEDDAR, A. A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 1159–1164. IEEE (2009).
- [42] FREDA, L. AND ORIOLO, G. Frontier-based probabilistic strategies for sensor-based exploration. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 3881–3887. IEEE (2005).
- [43] FRESE, U., LARSSON, P., AND DUCKETT, T. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, **21** (2005), 196.
- [44] FREUND, Y. AND SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, **55** (1997), 119.
- [45] GALINDO, C., SAFFIOTTI, A., CORADESCHI, S., BUSCHKA, P., FERNANDEZ-MADRIGAL, J.-A., AND GONZÁLEZ, J. Multi-hierarchical semantic maps for mobile robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2278–2283. IEEE (2005).
- [46] GAO, W., ZHANG, X., YANG, L., AND LIU, H. An improved sobel edge detection. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5, pp. 67–71. IEEE (2010).
- [47] GARCIA-GARCIA, A., ORTS-ESCOLANO, S., OPREA, S., VILLENA-MARTINEZ, V., AND GARCIA-RODRIGUEZ, J. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, (2017).
- [48] GEIGER, A., LENZ, P., AND URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361. IEEE (2012).
- [49] GELFAND, N., MITRA, N. J., GUIBAS, L. J., AND POTTMANN, H. Robust global registration. In *Symposium on geometry processing*, vol. 2, p. 5 (2005).
- [50] GREENE, N., KASS, M., AND MILLER, G. Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 231–238. ACM (1993).



- [51] GRISETTI, G., KÜMMERLE, R., AND NI, K. Robust optimization of factor graphs by using condensed measurements. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 581–588. IEEE (2012).
- [52] GRISETTI, G., KUMMERLE, R., STACHNISS, C., AND BURGARD, W. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, **2** (2010), 31.
- [53] GRISETTI, G., STACHNISS, C., GRZONKA, S., AND BURGARD, W. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems*, vol. 3, p. 9 (2007).
- [54] GUTMANN, J.-S. AND KONOLIGE, K. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pp. 318–325. IEEE (1999).
- [55] HANDA, A., WHELAN, T., McDONALD, J., AND DAVISON, A. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2014).
- [56] HARNAD, S. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, **42** (1990), 335.
- [57] HARRIS, C. AND STEPHENS, M. A combined corner and edge detector. In *Alvey vision conference*, vol. 15, pp. 10–5244. Citeseer (1988).
- [58] HARTLEY, R. AND ZISSERMAN, A. *Multiple view geometry in computer vision*. Cambridge university press (2003).
- [59] HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, **31** (2012), 647.
- [60] HERMANS, A., FLOROS, G., AND LEIBE, B. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2631–2638. IEEE (2014).
- [61] HERTZBERG, C., WAGNER, R., FRESE, U., AND SCHRÖDER, L. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, **14** (2013), 57.
- [62] HOLZ, D. AND BEHNKE, S. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Intelligent autonomous systems 12*, pp. 61–73. Springer (2013).
- [63] HOPPE, H., DEROSE, T., DUCHAMP, T., McDONALD, J., AND STUETZLE, W. *Surface reconstruction from unorganized points*, vol. 26. ACM (1992).

- [64] HORNING, A., WURM, K., BENNEWITZ, M., STACHNISS, C., AND BURGARD, W. OctoMap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, **34** (2013), 189. Available from: <http://www.informatik.uni-freiburg.de/~stachnis/pdf/hornung13auro.pdf>.
- [65] HOWARD, A., MATARIC, M. J., AND SUKHATME, G. Relaxation on a mesh: a formalism for generalized localization. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, pp. 1055–1060. IEEE (2001).
- [66] HULIK, R., BERAN, V., SPANEL, M., KRSEK, P., AND SMRZ, P. Fast and accurate plane segmentation in depth maps for indoor scenes. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1665–1670. IEEE (2012).
- [67] ILLINGWORTH, J. AND KITTLER, J. A survey of the hough transform. *Computer vision, graphics, and image processing*, **44** (1988), 87.
- [68] IRANI, S. AND RAGHAVAN, P. Combinatorial and experimental results for randomized point matching algorithms. *Computational Geometry*, **12** (1999), 17.
- [69] JOHNS, E., LEUTENEGGER, S., AND DAVISON, A. J. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3813–3822 (2016).
- [70] KAESS, M. Simultaneous Localization and Mapping with Infinite Planes. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2015).
- [71] KAESS, M., JOHANSSON, H., ROBERTS, R., ILA, V., LEONARD, J. J., AND DELLAERT, F. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, **31** (2012), 216.
- [72] KAESS, M., RANGANATHAN, A., AND DELLAERT, F. isam: Fast incremental smoothing and mapping with efficient data association. In *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1670–1677. IEEE (2007).
- [73] KERL, C., STURM, J., AND CREMERS, D. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 2100–2106. Citeseer (2013).
- [74] KERL, C., STURM, J., AND CREMERS, D. Robust odometry estimation for rgb-d cameras. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 3748–3754. IEEE (2013).
- [75] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *science*, **220** (1983), 671.

- [76] KLEIN, G. AND MURRAY, D. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234. IEEE (2007).
- [77] KLEIN, G. AND MURRAY, D. Improving the agility of keyframe-based slam. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 802–815. Springer (2008).
- [78] KLEIN, G. AND MURRAY, D. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pp. 83–86. IEEE (2009).
- [79] KOPPULA, H. S., ANAND, A., JOACHIMS, T., AND SAXENA, A. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in neural information processing systems*, pp. 244–252 (2011).
- [80] KRIEGEL, S., BRUCKER, M., MARTON, Z.-C., BODENMULLER, T., AND SUPPA, M. Combining object modeling and recognition for active scene exploration. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 2384–2391. IEEE (2013).
- [81] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105 (2012).
- [82] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105 (2012).
- [83] KÜMMERLE, R., GRISETTI, G., STRASDAT, H., KONOLIGE, K., AND BURGARD, W. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3607–3613. IEEE (2011).
- [84] LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).
- [85] LAVALLE, S. M. Rapidly-exploring random trees: A new tool for path planning. (1998).
- [86] LÁZARO, M. T., CAPOBIANCO, R., AND GRISETTI, G. Efficient long-term mapping in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2018).
- [87] LEMAIRE, T. AND LACROIX, S. Monocular-vision based slam using line segments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 2791–2796 (2007). doi:10.1109/ROBOT.2007.363894.
- [88] LI, X. AND GUSKOV, I. Multiscale features for approximate alignment of point-based surfaces. In *Symposium on geometry processing*, vol. 255, p. 217. Citeseer (2005).

- [89] LISIN, D. A., MATTAR, M. A., BLASCHKO, M. B., LEARNED-MILLER, E. G., AND BENFIELD, M. C. Combining local and global image features for object class recognition. In *Computer vision and pattern recognition-workshops, 2005. CVPR workshops. IEEE Computer society conference on*, pp. 47–47. IEEE (2005).
- [90] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., AND BERG, A. C. Ssd: Single shot multibox detector. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 21–37. Springer (2016).
- [91] LONG, J., SHEHMER, E., AND DARRELL, T. Fully convolutional networks for semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440 (2015).
- [92] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, **60** (2004), 91.
- [93] LU, F. AND MILIOS, E. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, **4** (1997), 333.
- [94] LU, F. AND MILIOS, E. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic systems*, **18** (1997), 249.
- [95] LU, Y. AND SONG, D. Robust rgb-d odometry using point and line features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pp. 3934–3942 (2015).
- [96] MA, L., KERL, C., STÜCKLER, J., AND CREMERS, D. CPA SLAM: Consistent Plane-Model Alignment for Direct RGB-D SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2016).
- [97] MAGNUSSON, M., LILIENTHAL, A., AND DUCKETT, T. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics (JFR)*, **24** (2007), 803.
- [98] MCCORMAC, J., HANDA, A., DAVISON, A., AND LEUTENEGGER, S. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4628–4635. IEEE (2017).
- [99] MORGENSTERN, O. AND VON NEUMANN, J. *Theory of games and economic behavior*. Princeton university press (1953).
- [100] MUR-ARTAL, R. AND TARDÓS, J. D. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *arXiv preprint arXiv:1610.06475*, (2016).
- [101] NARDI, F., LÁZARO, M. T., IOCCHI, L., AND GRISETTI, G. Generation of laser-quality 2d navigation maps from rgb-d sensors. In *RoboCup Symposium 2018* (2018).

- [102] NEWCOMBE, R. A., LOVEGROVE, S. J., AND DAVISON, A. J. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327. IEEE (2011).
- [103] NEWCOMBE, R. A., ET AL. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 127–136 (2011). Available from: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf>.
- [104] NI, K., STEEDLY, D., AND DELLAERT, F. Tectonic slam: Exact, out-of-core, submap-based slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1678–1685. IEEE (2007).
- [105] NICHOLSON, L. J., MILFORD, M. J., AND SUNDERHAUF, N. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, (2018).
- [106] NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, **32** (2013), 169.
- [107] NÜCHTER, A. AND HERTZBERG, J. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, **56** (2008), 915.
- [108] NUCHTER, A., SURMANN, H., LINGEMANN, K., HERTZBERG, J., AND THRUN, S. 6d slam with an application in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2, pp. 1998–2003. IEEE (2004).
- [109] OEHLER, B., STUECKLER, J., WELLE, J., SCHULZ, D., AND BEHNKE, S. Efficient multi-resolution plane segmentation of 3d point clouds. In *International Conference on Intelligent Robotics and Applications*, pp. 145–156. Springer (2011).
- [110] OHLANDER, R., PRICE, K., AND REDDY, D. R. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, **8** (1978), 313.
- [111] OLIVA, A. AND TORRALBA, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Intl. Journal of Computer Vision (IJCV)*, **42** (2001), 145.
- [112] OLSON, E., LEONARD, J., AND TELLER, S. Fast iterative alignment of pose graphs with poor initial estimates. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2262–2269. IEEE (2006).
- [113] OSHER, S. AND SETHIAN, J. A. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, **79** (1988), 12.

- [114] PAPAGEORGIOU, C. P., OREN, M., AND POGGIO, T. A general framework for object detection. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pp. 555–562. IEEE (1998).
- [115] PAPAIOV, C. AND BURSCHEKA, D. Stochastic optimization for rigid point set registration. In *International Symposium on Visual Computing*, pp. 1043–1054. Springer (2009).
- [116] PATHAK, K., BIRK, A., VASKEVICIUS, N., AND POPPINGA, J. Fast registration based on noisy planes with unknown correspondences for 3-d mapping. *IEEE Transactions on Robotics*, **26** (2010), 424.
- [117] POMERLEAU, F., COLAS, F., SIEGWART, R., ET AL. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, **4** (2015), 1.
- [118] POMERLEAU, F., MAGNENAT, S., COLAS, F., LIU, M., AND SIEGWART, R. Tracking a depth camera: Parameter exploration for fast icp. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3824–3829. IEEE (2011).
- [119] POPPINGA, J., VASKEVICIUS, N., BIRK, A., AND PATHAK, K. Fast plane detection and polygonalization in noisy 3d range images. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3378–3383. IEEE (2008).
- [120] POTTHAST, C. AND SUKHATME, G. S. Next best view estimation with eye in hand camera. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2011).
- [121] PRONOBIS, A. AND JENSFELT, P. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3515–3522. IEEE (2012).
- [122] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788 (2016).
- [123] RODOLÀ, E., ALBARELLI, A., BERGAMASCO, F., AND TORSSELLO, A. A scale independent selection process for 3d object recognition in cluttered scenes. *International journal of computer vision*, **102** (2013), 129.
- [124] ROSTEN, E. AND DRUMMOND, T. Machine learning for high-speed corner detection. In *European conference on computer vision*, pp. 430–443. Springer (2006).
- [125] ROSTEN, E. AND DRUMMOND, T. Machine learning for high-speed corner detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 430–443 (2006).

- [126] RUIZ-SARMIENTO, J.-R., GALINDO, C., AND GONZALEZ-JIMENEZ, J. Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems*, **119** (2017), 257.
- [127] RUSINKIEWICZ, S. AND LEVOY, M. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152. IEEE (2001).
- [128] RUSU, R. B., BLODOW, N., AND BEETZ, M. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3212–3217. Citeseer (2009).
- [129] RUSU, R. B. AND COUSINS, S. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China (2011).
- [130] RUSU, R. B., MARTON, Z. C., BLODOW, N., DOLHA, M., AND BEETZ, M. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, **56** (2008), 927.
- [131] SAAD, Y. *Iterative methods for sparse linear systems*. SIAM (2003).
- [132] SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., KELLY, P. H., AND DAVISON, A. J. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1352–1359. IEEE (2013).
- [133] SCHLEGEL, D. AND GRISETTI, G. HBST: A hamming distance embedding binary search tree for visual place recognition. *CoRR*, **abs/1802.09261** (2018). Available from: <http://arxiv.org/abs/1802.09261>, arXiv:1802.09261.
- [134] SCHNABEL, R., WAHL, R., AND KLEIN, R. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, vol. 26, pp. 214–226. Wiley Online Library (2007).
- [135] SEGAL, A., HAEHNEL, D., AND THRUN, S. Generalized-icp. In *Proc. of Robotics: Science and Systems (RSS)*, vol. 2 (2009).
- [136] SENARATHNE, P., WANG, D., WANG, Z., AND CHEN, Q. Efficient frontier detection and management for robot exploration. In *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*, pp. 114–119. IEEE (2013).
- [137] SERAFIN, J. AND GRISETTI, G. Using extended measurements and scene merging for efficient and robust point cloud registration. *Robotics and Autonomous Systems*, **92** (2017), 91.
- [138] SERAFIN, J. AND GRISETTI, G. Using extended measurements and scene merging for efficient and robust point cloud registration. *Journal on Robotics and Autonomous Systems (RAS)*, **92** (2017), 91. Available from: <https://doi.org/10.1016/j.robot.2017.03.008>, doi:10.1016/j.robot.2017.03.008.

- [139] SHI, J. AND MALIK, J. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **22** (2000), 888.
- [140] SIMONYAN, K. AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014).
- [141] SMITH, P., REID, I. D., AND DAVISON, A. J. Real-time monocular slam with straight lines. *Proc. of British Machine Vision Conference (BMVC)*, (2006).
- [142] STAMPFER, D., LUTZ, M., AND SCHLEGEL, C. Information driven sensor placement for robust active object recognition based on multiple views. In *Technologies for Practical Robot Applications (TePRA), 2012 IEEE International Conference on*, pp. 133–138. IEEE (2012).
- [143] STEDER, B., RUSU, R. B., KONOLIGE, K., AND BURGARD, W. Point feature extraction on 3d range scans taking into account object boundaries. In *Robotics and automation (icra), 2011 IEEE international conference on*, pp. 2601–2608. IEEE (2011).
- [144] STEINBRÜCKER, F., STURM, J., AND CREMERS, D. Real-time visual odometry from dense rgb-d images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 719–722. IEEE (2011).
- [145] STÜCKLER, J., BIRESEV, N., AND BEHNKE, S. Semantic mapping using object-class segmentation of rgb-d images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 3005–3010. IEEE (2012).
- [146] STURM, J., ENGELHARD, N., ENDRES, F., BURGARD, W., AND CREMERS, D. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2012).
- [147] SÜNDERHAUF, N., DAYOUB, F., MCMAHON, S., TALBOT, B., SCHULZ, R., CORKE, P., WYETH, G., UPCROFT, B., AND MILFORD, M. Place categorization and semantic mapping on a mobile robot. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 5729–5736. IEEE (2016).
- [148] SWAIN, M. J. AND BALLARD, D. H. Color indexing. *Intl. Journal of Computer Vision (IJCV)*, **7** (1991), 11.
- [149] SZEGEDY, C., ET AL. Going deeper with convolutions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2015).
- [150] TABIB, W., O’MEADHRA, C., AND MICHAEL, N. On-manifold gmm registration. *IEEE Robotics and Automation Letters*, **3** (2018), 3805.
- [151] TAGUCHI, Y., JIAN, Y. D., RAMALINGAM, S., AND FENG, C. Point-plane slam for hand-held 3d sensors. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 5182–5189 (2013).



- [152] TAM, G. K., CHENG, Z.-Q., LAI, Y.-K., LANGBEIN, F. C., LIU, Y., MARSHALL, D., MARTIN, R. R., SUN, X.-F., AND ROSIN, P. L. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, **19** (2013), 1199.
- [153] TENORTH, M., KUNZE, L., JAIN, D., AND BEETZ, M. Knowrob-map-knowledge-linked semantic object maps. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pp. 430–435. IEEE (2010).
- [154] TOMASI, C. AND KANADE, T. Detection and tracking of point features. (1991).
- [155] TORABI, L. AND GUPTA, K. An autonomous six-dof eye-in-hand system for in situ 3d object modeling. *The International Journal of Robotics Research*, **31** (2012), 82.
- [156] TORRALBA, A., MURPHY, K. P., FREEMAN, W. T., AND RUBIN, M. A. Context-based vision system for place and object recognition. In *null*, p. 273. IEEE (2003).
- [157] TREVOR, A. J., GEDIKLI, S., RUSU, R. B., AND CHRISTENSEN, H. I. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, (2013).
- [158] TREVOR, A. J., ROGERS, J. G., AND CHRISTENSEN, H. I. Planar surface slam with 3d and 2d sensors. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 3041–3048. IEEE (2012).
- [159] TUYTELAARS, T., MIKOLAJCZYK, K., ET AL. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, **3** (2008), 177.
- [160] ULRICH, I. AND NOURBAKHSI, I. Appearance-based place recognition for topological localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, vol. 2, pp. 1023–1029. Ieee (2000).
- [161] VALENTIN, J. P., SENGUPTA, S., WARRELL, J., SHAHROKNI, A., AND TORR, P. H. Mesh based semantic modelling for indoor and outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2067–2074 (2013).
- [162] VASQUEZ-GOMEZ, J. I., SUCAR, L. E., AND MURRIETA-CID, R. View planning for 3d object reconstruction with a mobile manipulator robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4227–4233. IEEE (2014).
- [163] VINEET, V., ET AL. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 75–82. IEEE (2015).
- [164] VIOLA, P. AND JONES, M. J. Robust real-time face detection. *Intl. Journal of Computer Vision (IJCV)*, **57** (2004), 137.

- [165] VON GIOI, R. G., JAKUBOWICZ, J., MOREL, J.-M., AND RANDALL, G. On straight line segment detection. *Journal of Mathematical Imaging and Vision*, **32** (2008), 313.
- [166] VON GIOI, R. G., JAKUBOWICZ, J., MOREL, J.-M., AND RANDALL, G. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, **32** (2010), 722.
- [167] VON GIOI, R. G., JAKUBOWICZ, J., MOREL, J. M., AND RANDALL, G. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **32** (2010), 722. doi:10.1109/TPAMI.2008.300.
- [168] WANG, Y., LIANG, A., AND GUAN, H. Frontier-based multi-robot map exploration using particle swarm optimization. In *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, pp. 1–6. IEEE (2011).
- [169] WHELAN, T., KAESS, M., FALLON, M., JOHANNSSON, H., LEONARD, J., AND MCDONALD, J. Kintinuous: Spatially extended kinectfusion. (2012).
- [170] WHELAN, T., LEUTENEGGER, S., SALAS-MORENO, R., GLOCKER, B., AND DAVISON, A. Elasticfusion: Dense slam without a pose graph. In *Proc. of Robotics: Science and Systems (RSS)* (2015).
- [171] WHELAN, T., SALAS-MORENO, R. F., GLOCKER, B., DAVISON, A. J., AND LEUTENEGGER, S. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, **35** (2016), 1697.
- [172] WU, K., RANASINGHE, R., AND DISSANAYAKE, G. Active recognition and pose estimation of household objects in clutter. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4230–4237. IEEE (2015).
- [173] ZENDER, H., MOZOS, O. M., JENSFELT, P., KRUIJFF, G.-J., AND BURGARD, W. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, **56** (2008), 493.
- [174] ZENG, M., ZHAO, F., ZHENG, J., AND LIU, X. Octree-based fusion for realtime 3d reconstruction. *Graphical Models*, **75** (2013), 126.
- [175] ZHANG, J. AND SINGH, S. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2174–2181. IEEE (2015).
- [176] ZHANG, L. AND KOCH, R. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Visual Communication and Image Representation*, (2013), 794.
- [177] ZHANG, Z. *Iterative point matching for registration of free-form curves*. Ph.D. thesis, Inria (1992).