UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXVI

Robust and Efficient Geometric Methods for Registration, Localization and Mapping

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Dario Lodi Rizzini

Dottorando: Ernesto Fontana

Anni 2020-2023

To my family.
My past, my present, and my future.

The journey through my three PhD Years has been quite a run. Starting in the middle of the pandemic has made the accommodation process a little harder than usual. The remote work has somewhat impacted my motivation, as I was still feeling more of a student continuing its Master's thesis, rather than a researcher. Luckily, all of this started turning around at the end of the first year, with the social element of meeting so many new and amazing people being an essential factor that had been missing prior. Summer schools, conferences, project meetings, and other gatherings have helped me feeling part of a research community where people struggle and rejoice for similar reasons to mine. First, I want to thank my advisor Dario, for all the help and support he managed to constantly provide, together with the other RIMLab colleagues. Then, an important mention goes to Martina, for the everlasting way in which she stands by my side, especially during my period abroad in the US. Also, I want to thank all the new friends I have encountered. I hope that we will all manage to continue on with successful and rewarding careers. Finally, a huge thought goes to my spread-out family. Without forgetting the *evolving* Italian side, a big shout out goes to Gil, Lindsey and little Ale, as they have always had my back, and have guided me during the most important steps of my visiting period. And of course, last but not least, all of this would have been way harder without the loving eyes of my parents watching my every step.

Contents

Chapter 1

Introduction

In the last couple of decades, robotics has developed vertically towards a growing number of fields. Nowadays, robotics engineering encompasses a wide set of very diverse jobs, spanning from the strictly electrical or mechanical ones, to multi-robot fleet coordination, or high-level software development. The rise among the computer science community of the *artificial intelligence* techniques has caused this trend to accelerate even further. This is due to the tendency of applying machine learning techniques to an ever increasing number of fields. Still, many industrial robotics tasks are performed today through solutions analogous to the ones they were first introduced to the market with, many years ago. For this relevant number of use cases, it is important to keep research going on improving efficiency and diversity of robotic *primitives*. Primitives can be defined as tasks that are relevant in a wide number of applications. There are no clear rules for defining what can be considered as a primitive. For example, a detection algorithm that is applicable to different sensory data can be considered as a primitive, but the line becomes greyer when thinking about algorithms that only work on a category of sensors. Moreover, many robotic applications require accurate and formally guaranteed estimation algorithms. While data-driven approaches enable addressing the variety of empirical conditions, accurate robot positioning and navigation, as well as sensor-driven applications in safety-critical scenarios, rely on geometry-based mathematical models. Anyway, despite research in robotics getting

increasingly focused on very specific tasks, being able to understand and continuatively improve the very primitives used by most robotic systems has great importance. The registration primitives tackled in this dissertation are based on geometric features and are able to estimate transformations that can be described through the use of Lie algebra and Lie groups in particular [1]. Also, the most sophisticated approaches to optimization presented here make use of the manifold interpretation of these groups, that is typical of Riemannian Geometry. The SLAM (Simultaneous Localization and Mapping) problem can be seen as a series of subproblems that, when holistically organized within a system, enable a robot to perform actions in its working space, with the needed awareness of the surroundings. In this context, the registration problem can be seen as one of the cornerstones for performing solid SLAM. The aforementioned task of registration consists in aligning partially overlapping data, for example point clouds, which are discrete sets of data points in space, possibly containing additional information such as color, intensity and confidence level. For spatial data (e.g., 2D or 3D point clouds, with points described through their Cartesian coordinates), the most common type of registration to be performed is the estimation of a rigid transformation that links them. An easily understandable use case for this rigid transformation estimation task is the one of a mobile robot that acquires sensory data of its surroundings, and needs to link them in space, in order to construct a map, or simply to figure out its position in the environment after losing track of the odometry. The rigid transformation estimation consists in estimating a rigid pose, which is well known to be decomposable in the two parts of rotation and translation. After decades of research, several estimation algorithms and primitives have been developed to address the robot localization and mapping problem and, in particular, point cloud registration. Most registration algorithms function by minimizing an objective function that formalizes the rigid transformation separating the input data. These optimizations are often done through local searches, with assumptions that enable avoiding the use of more robust methods e.g., being able to register only small transformations. Although these methods definitely present a good number of use cases, a general formulation should present a certain robustness, enabling it to work with as few hypotheses as possible. The concept of robustness is given clearer shape

when problems are being tackled as optimization problems: robust methods should be able to reach and output the global optimum of the problem's cost function. Then, operations research theory can be used to provide guarantees on the ability of an algorithm to constantly produce the correct result. Such guarantees assume essential importance, and are ever-more relevant in critical applications. However, most of the time removing assumptions from a problem can lead to an increased computational complexity of the algorithms that solve it globally. Because of this, it is paramount to keep an eye on how to reduce computational complexity when following such approaches e.g., simplifying input data by eliminating redundancies, or parallelizing as many operations as possible.

In general, pose estimation is achieved through a single optimization on the whole transformation. However, as most of the times the hardest task of a registration problem is the rotation estimation, it can be a good idea to decouple it from translation estimation, in order to properly focus more on the heavier part between the two. When this is possible, the goal becomes to correctly estimate rotation independently from translation, and estimate translation on an already rotated set of points only later. A series of works based on this approach is presented in Chapters 3, 4 and 5.

As previously said, the registration problem can be used as a stepping stone towards robot localization. The localization task allows a robot to determine its position in the environment. It assumes great importance in several cases, especially when it has to be performed among multiple robots that cooperate in a fleet (e.g., of drones). From rapid deliveries of critical items for middle-distances, to their use in agriculture or even in the military, the coordination of drones forming a fleet has seen a rapid growth in importance. This problem can be seen as a case of the multi-agent robotics paradigm and be solved accordingly. An innovative approach to the solution of this problem without a pre-definite master drone, and based on the Shape of Motion (acronym SOM) formulation, is presented in Chapter 6.

1.1 Contributions

In this dissertation, I present the following contributions:

- The general definition of Angular Radon Spectrum (ARS) of a GMM representing a point cloud in d-dimension space and the proof of its translation-invariance and rotation-shift property.
- A computation procedure of anisotropic ARS in 2D domain with GMM simplification for rotation estimation.
- The parallel algorithm Cud-ARS for computation of the Fourier coefficients of ARS suitable for parallel execution of GPUs.
- The derivation of closed-form spherical harmonics expansion of ARS in 3D domain and an algorithm for 3D rotation estimation.
- The formulation of the Cross-Localization through Shape of Motion (SOM) problem and its solution based on modified Riemannian Staircase.

1.2 Document Organization

After present Introduction chapter, this dissertation (name chosen over the less common *thesis*) proceeds as follows. Chapter 2 presents the state of the art of registration and localization problems to provide the necessary context and reasoning which led to the choice of the approaches presented in the next chapters. Chapter 3 provides a general introduction of the Angular Radon Spectrum (ARS) paradigm in the most general d-dimensional case; one section is dedicated to the presentation of a translation estimation algorithm based on branch-and-bound optimization, that enables full pose estimation. Chapter 4 goes more into the details of the planar version of ARS. Chapter 5 instead focuses on the spatial version of ARS. Chapter 6 contains the presentation of a novel pose averaging method based on the Shape of Motion (acronym SOM) formulation. Finally, Chapter 7 provides analysis and comment of the results, together with the final remarks.

Experimental results are discussed after each relevant theoretical section that introduces new concepts that need testing.

Chapter 2

State of the Art

2.1 Point Cloud Registration

In recent years, robotics research has seen rapid but substantial changes. First, a great number of affine fields have risen in terms of importance and are now almost considered as integral part of the wide robotics field. For example, sensory data elaboration has introduced a number of new techniques that reflect the development of sensor technology. Nowadays it is relatively common to have sensors able to acquire millions of data points with high frequency, and very specific and efficient algorithms for their elaboration are required. The necessity of a very vertical knowledge in an increasing number of fields is apparent today. One traditionally widely used tool for representing sensory data is the *point cloud*. Point clouds represent objects and scenes through a collection of points corresponding to sensor measurements. Points are essentially represented by their Cartesian coordinates, although additional properties can be included, such as color, surface normal, sensor viewpoint, signal intensity and other types of information that can be used for specific formulations. However, the geometric information given by the point coordinates is the most general and, as such, is present in the vast majority of point clouds. In the most general sense, point clouds are a means of collating a large number of single spatial measurements into a dataset that can then represent a whole scene. Two main problem categories are

analyzed in the context of this dissertation. The first problem is the Simultaneous Localization and Mapping (acronym SLAM), with a particular focus on its registration subtask. The registration problem aims at aligning partially overlapping data that are separated from one another by various degrees of freedom: translation (3 degrees of freedom), rotation (3DOFs), pose (6DOFs), pose + scale (7DOFs) and so on. Over the years, multiple solutions to registration have been proposed, covering different use cases that cover a vast taxonomy of approaches:

- Global and local registration.
  Local registration algorithms use only neighborhood information of the cloud points, while global registration methods take into account the entire point clouds. The trade-off between an increased precision, but a potentially higher computational cost for the global methods is apparent.

- Planar and spatial registration.
  Registration methods are sometimes formulated in a specific way for planar point clouds (e.g., laser scans), while other are more general.

- Correspondence-based and correspondence-less registration.
  Registration methods can be reliant on the estimation of correspondences [2] (e.g., through features such as normal estimation, patch detection, FPFH [3], neighbor analysis etc.) and be performed on the features themselves, or otherwise act directly on the point clouds without any pre-processing.

- Formal analysis and certifiability of solution.
  A registration algorithm can be formulated as an optimization problem and, as such, its optimality and bound conditions can be analyzed and offered as proof of the method's validity. Additionally, the formal description of the registration function can be based on either discrete or continuous mathematics.

However, this taxonomy is not fully orthogonal, and some categories can often be found together in some method classes. For example, correspondence-based algorithms are usually local algorithms, since correspondences between point cloud parts

are established using a given an initial guess. Also, the approaches to solving problems have undergone a widely known paradigm shift. Led by the massive increase in data availability, artificial intelligence-based techniques have seen great success, especially in detection or recognition tasks.

2.1.1 Local Methods

The alignment of point clouds is an important primitive for many applications in robot localization and mapping. Registration also allows merging of views of the same object or scene differing in orientation and position. The standard approach to registration requires detection of corresponding parts between the two views, which are usually represented in the form of point clouds. Correspondence-based methods [4, 5, 6] estimate the rigid transformation by minimizing the distances between corresponding points in the two compared point sets. The computation of rigid transformation has well known closed-form solution for 2D, 3D and arbitrary dimensions, but in order to establish correct point-to-point or point-to-surface associations a good initial guess of the transformation is required. Inaccurate correspondences often lead to wrong estimation, since standard registration methods only solve local optimization problems.

In this context, classical methods are still relevant today, but need some additional contextualization. The Iterative Closest Point (ICP) [7, 8] paradigm has been used and reformulated multiple times since its first appearance in the early 1990's, and is probably the most well know approach to registration. The critical step of ICP is the search for correct matches between points, whereas the computation of the best rigid transformation is a solved problem both for 2D and 3D cases.

Another classic approach to registration in 2D domain uses correlation of occupancy grid map [9], which is effective once the discretization parameters have been properly tuned. This approach estimates the transformation between point clouds though global optimization of objectives functions depending on the features. Then, the feature viewpoint invariance is a major factor. Several algorithms allow decoupling of rotation and translation estimation thanks to independence of the features.

One important discussion to be made for non-global methods regards the avail-

ability of initial guess. This is usually retrieved by another measurement (e.g. the robot odometry) or by preliminary feature matching [6]. Robust associations techniques like Vector Field Consensus (VFC) [5] have been proposed to detect consistent associations and filter outliers associations. In this context come recent works from Vizzo, Guadagnino, Stachniss *et al.* [10, 11], which aim to refurbish traditional approaches to LOAM (Lidar Odometry and Mapping) and ICP. Nowadays, multi-layer Lidars have become popular for several applications, including autonomous driving and industrial applications. These sensors acquire high range accurate geometric measurements and provide accurate representation of 3D scene occupancy with a single scan in the form of point clouds. Due to this, a specialized research branch has developed methods and algorithms that rely on the specific data structure that Lidars output. The main limitation of this type of sensor lies in non-uniform angular resolution and field-of-view (FoV) of range measurements, which are vertically sparse and horizontally dense. The large amount of data collected is also a challenge for online computations like point set registration, sensor odometry and mapping. Specialized algorithms have been proposed for the sparse point clouds acquired by multi-layer Lidars. An effective approach relies on extraction of salient features that facilitates association and registration. LOAM [12] is among the first effective algorithms using crafted keypoints for pairing consecutive points. Several improved versions of the algorithm have been proposed to operate with other sensors. For example, F-LOAM (Fast LOAM) [13] is an efficient and effective system that requires only 3D Lidar measurements. It extends the original work by combining the geometric and images data, working on detection of high curvature points (edge) and planar patches (surface) from point clouds organized in different layers. Other registration and mapping tools such as LIO (Lidar Inertial Odometry) [14] and IN2LAMA (INertial Lidar Localization And MApping) [15] are based on similar features.

2.1.2 Global and Robust Methods

A relevant variation of ICP that renders able to search the *global* optimum is GO-ICP by [16]. This is a branch-and-bound (B&B) based optimization algorithm that is able to solve the pose registration task in $SE(3)$. Despite its accuracy being surpris-

ing, GO-ICP can suffer from an exponential growth in time complexity for it resolution, leading to very long estimation times even for relatively simple cases. Several correspondence-less methods enable separation between estimation of rotation and of translation. Global registration algorithms often rely on features like local descriptors, global histograms or functions extracted from each point set. However, in general they are not adequately efficient for practical applications. Several correspondence-less methods enable separation between estimation of rotation and of translation. They include Hough Spectrum (HS) [17, 18] and Angular Radon Spectrum (ARS) [19], Fourier and spherical harmonics expansion of point distributions [20]. Another relevant global registration method that makes use Fourier and spherical harmonics expansion of point is Phaser [21] from Bernreiter *et al.* . One trend that multiple authors have shown interest in, is the reformulation of classical approaches in order to adapt them to the new trends and use cases of modern robotics. A series of works from MIT authors [22, 23, 24] is aimed at proposing and exploiting TEASER, a certifiable and powerful outlier rejection, that can also be used for estimating 7DOF poses in limited cases. Formally guaranteed methods like TEASER effectively estimate with high number of outlier associations, but they may still fail with highly inaccurate initial correspondences. They exploit regularization and other consensus techniques to remove outlier associations. Also, the time complexity of these algorithms grows considerably fast, even with only $\sim$100 correspondences to register. Another category of global methods model the input point clouds as a continuous functions like a Bayesian model of normals [25] or kernel Hilbert space [26, 27] and perform global optimization.

2.1.3 Deep Learning Methods

A series of articles, including prominent authors from the early (late '90s-early 2000s) SLAM community [28, 29] reflects on the actuality of the SLAM paradigm and on the potential, as well as the limits, of deep learning (DL) and convolutional neural network (CNN) techniques. Despite the goal of these articles not being the one of giving clear directions moving forward, but more the one of facilitating discussions and bring clarity to the state of the art at the time, some general opinions can still be

gathered.

Still, one of the great challenges that appears to be unsolved is finding a soft spot for mixing the more traditional approaches with the novel and more recent ones. An increasing number of recent works has focused on certifying optimality, or at least stability/boundability of the results obtained through DL-based approaches [30]. This is an important trend also in industrial robotics, with many companies worldwide exploring the capabilities of deep learning techniques in a growing number of applications, but still somewhat limiting their use in critical applications due to the lack of formal guarantees of most CNN-based methods. Aoki *et al.* have recently proposed PointNetLK [31]. PointNetLK applies the Lucas-Kanade [32] algorithm by Baker and Matthews [33], and a more recent adaptation of it [34] to the object tracking task, to the registration problem. It is one of the first works to successfully tackle point cloud registration through a deep learning approach (specifically, using a descriptor based on PointNet [35]). It performs global registration in the sense that the loss function to be minimized through PointNet uses global features. However, its computational cost is still high for real-time applications, and the deep learning approach is naturally reliant on the quality of the training. A more recent work [36] even proposed an analytical improvement over for the feature update of PointNetLK.

Using another deep learning based approach, Kurobe *et al.* proposed CorsNet [37], which directly aims at solving registration through Deep Neural Networks (DNNs). A deep-learning based approach with similar bases to the one discussed in this dissertation is DeepGMR by Yuan *et al.* [38]. A series of works from Wang *et al.* [39, 40] tackles multiple aspects of the DL-based registration, first analyzing and proposing point cloud representations that are suitable for training, and then using them inside a partial-to-partial registration pipeline. A relevant aspect is the choice of features selected for training. this particular case, they are geometric features extracted through KeypointNet [41]. These allow the method to be considered as *unsupervised*, as they are extracted without additional labeling.

2.1.4 Datasets and Benchmarks

Already more than ten years ago, Pomerleau *et al.* [42] analyzed the problem of establishing a solid and challenging benchmark for testing registration performances, but today the importance Especially with the rise of deep learning based methods, the importance of having huge but still usable datasets has grown considerably. A good number of works sees a major contribution in acquiring, labeling, and publishing datasets. Smaller, but very useful datasets can also be found for specific applications.

In-depth surveys for registration in specific cases can also be found. An example can be [43] by Pomerleau *et al.* that focuses heavily on mobile robots and many ICP variations. Recently, Zhang *et al.* [44] wrote an article that analyzes tendencies, strength and limitations of the growing use of deep learning for the registration task. While listing and describing many notable algorithms, they make relevant points of their dissertation focusing on the usage of features and their matching, as well as on refinement, outlier rejection, and making comparisons with traditional registration techniques. Also, a full chapter is dedicated to listing relevant datasets and their features.

2.1.5 GPU-Based Parallel and Data-Intensive Methods

Another trend that has received an increasing popularity with the development of data-intensive techniques is the formulation of highly parallelizable algorithms to be run on the GPUs (Graphics Processing Units). Mathematical frameworks to deal with similar problems on the GPU have already been proposed [45, 46, 47], but they tend to lack when it comes to implementation-wise guidance. Despite the usage of SLAM methods in real-time applications being a long time concern [48, 49], the exploitation of GPUs to speed-up perception and sensor processing [50] is less frequent in literature. It is instead more common to see it paired with some computer vision primitives or straight-up deep learning methods [51, 52, 53]. As a matter of fact, GPUs are generally more used in approaches that feature some level of computer vision into them, or even straight-up CNNs [51]. There are some examples of proposed integration between robotics and GPU computation. Milioto *et al.* proposed

Bonnet [54] and RangeNet++ [55] for segmentation based on deep learning. The first is a framework based on different state-of-the-art CNN implementations, enabling multi-GPU simplified training, as well as further expansion of the framework itself. The latter is a LIDAR-specific segmentation method, optimized for high-speed processing in order for it to be usable for autonomous driving applications. One work that goes in a similar direction to ours is Collet *et al.* proposed a series of works leading to MOPED [56, 57], a framework that at its core is able to estimate the pose of objects from recognizing feature keypoints. A typical application for some of these frameworks is robotic manipulation [58]. Furthermore, Titan [59] is a library comprising parallel algorithms to handle geometry in soft-body and multi-robots physics simulations. Such approach to parallel processing with Nvidia CUDA closely resembles the proposed operational decomposition of ARS. Even Nvidia themselves have recently introduced the ISAAC platform [60] and is producing a good number of robotic works, spanning from manipulation and grasping all the way to detection and registration. GPU-related literature also includes a class of works focusing on mathematical-wise considerations regarding computational optimization and parallelization. For example, Ha *et al.* [61] present an optimal parallel scan method, showing experiments on throughput and MIPS on a data-intensive simulation of a prefix sum problem. Still, a fillable gap between parallelization analysis of benchmark problems and deep learning-related applications, specifically in the field of robotic registration and mapping, appears to exist to this day.

2.2 Multi-Robot Pose Estimation

Distributed robotics systems have been relevant in robotics research ever since the 1980s [62]. Naturally being in constant evolution during all these years due to the ever-changing (intra-fleet but inter-robots) communication techniques (e.g., cellular technology [63], bluetooth, Wi-Fi, infrared, etc.), the field of distributed robotics is one of the most challenging in robotics due to the significant growth in complexity that separates a single robot from a fleet-based robotic application [64, 65]. As a matter of fact, distributed robotics system need to have seamless integration in the motion

planning of each robot within the system [66], and be able to execute a number of simultaneous actions equal to the number of robots present in the system.

2.2.1 UAV Use Case

Probably, as of today, the most relevant use case of this setup is a fleet of unmanned aerial vehicles (UAVs e.g., drones). UAVs are used for a vast number of purposes, from retail sale delivery (with a very rapid development in the last eight years, also due to the COVID-19 pandemic from feasibility studies [67, 68] to already improvement and optimization of systems [69, 70, 71, 72]), to critical and man-unfriendly quality control applications (e.g., pipeline monitoring [73]), all the way to the military [74]. A problem that has been investigated in this context by other research fields is bundle adjustment [75, 76].

2.2.2 Intra-Fleet Cross-Localization

One of the most typical problems that a distributed robotics system has to confront itself with is the reciprocal localization of each robot inside the system. The problem of distributed robotics to be analyzed that is most interesting in the context of this dissertation is the one of *pose averaging* [77, 78, 79]. Multiple pose averaging is a methods of estimating camera fields-of-view (FoVs) using the information available from cameras in the network [80]. In other words, the problem aims to obtaining the set of absolute camera poses (rotation and translation) given a set of noisy camera measurements/estimations. It has been used for decades in Structure-from-Motion reconstructions. This problem encompasses a wide number of cases, starting from the single rotation averaging [81, 82]. The theory and proposed approaches to solve the multiple pose averaging problem can be used to perform the cross-localization of a set of cameras, given a partial and noisy set of the translations between one another.

2.2.3 Relation with SLAM

Registration and localization algorithms have been used for multiple purposes, in both robotics and other fields like computer vision. The already mentioned SLAM

problem can encompass these tasks, along with mapping and many others as part of a more complex system. Many solutions for SLAM have been proposed over the years, and research for their improvement continues also today. In particular, the graphical SLAM formulation [83, 84, 85] has received a good amount of attention over the years, due to its ability to pair flexibility with formal soundness. However, many of the proposed solutions are limited in the strength of their ability to produce optimal solutions, as their reliance upon local search techniques leaves them vulnerable to convergence to significantly suboptimal critical points.

2.2.4 Certifiable Methods

A relevant work in the context of the robust approaches presented in this dissertation is SE-Sync [86] from Rosen *et al.* . In this work, the authors propose a certifiably correct algorithm for synchronization over the $SE(d)$ Lie group that have to be estimated given noisy measurements of a subset of their pairwise relative transforms. Their goal is to find a soft spot between the rapidity of local search [87, 88, 89, 90], other previous approaches through constraint relaxations [91, 92, 93, 87, 94, 95, 96, 97], while still being able to offer some sort of certification of the algorithm's output. SE-Sync is an algorithm able to provide a non-approximated certifiable solution, through the usage of semidefinite relaxations and the Riemannian Staircase paradigm (that will be analyzed in depth in Chapter 6). The certification of the solution is based on verifying the correctness of a reformulation of the initial maximum-likelihood problem. The latter is based on iteratively increasing the dimension of the solution search space, in order to exit local minima and provide more accurate solutions.

Chapter 3

ARS - Angular Radon Spectrum: General Definition and Introduction

Registration is the problem of finding the rigid transformation (i.e., rotation and translation) that better aligns two or more views of the same object or scene. It is an instance of the more general pose estimation problem, which involves partially, but substantially, overlapping representations. In general, pose estimation can be formulated in the form of networks of reference frames where pairwise relative measurements are given. Registration can be seen as a primitive for general localization and mapping estimation. The problem tackled in this chapter is the estimation of a rigid motion/pose (i.e., roto-translation, 6 DOFs) that links a source and a destination point sets. Multiple use cases for this application can be though of, but the most appropriate would be a mobile robot navigating in a partially unknown environment which has to align the point clouds it is acquiring during its motion, through one or more sensors. The Angular Radon Spectrum (ARS) [19, 98, 99] paradigm has a notable importance for solving the 6 DOFs pose estimation between a pair of overlapping point clouds. To give a quick overview of its most important characteristics, ARS is a descriptor that can be used to robustly assess the rotation between two point clouds representing

the same scene from different viewpoints. Quickly recapping some concepts introduced in Chapter 2, the most common point cloud registration algorithms rely on the assessment of correspondences between points or regions of the point clouds to be aligned. The estimation of correspondences is often inferred starting from an initial guess of the pose to be estimated. The estimation of such pose is then iteratively refined by minimizing a cost function that measures the differences between the shapes. However, this procedure usually only allows search of local optima. This context is needed in order to understand the motivations behind one of the most important contributions presented in this dissertation. In particular, Chapters 3, 4, 5 contain the proposal and derivation of *Angular Radon Spectrum* (ARS), a robust feature able to strongly characterize the orientation of point clouds. ARS is based on a variation of the Radon Spectrum that produces a global descriptor containing relevant information regarding the collinearity (if the point cloud is planar) or coplanarity (if spatial) of points. Thus, it is an effective solution for addressing the difficult part of the registration problem. As a matter of fact, one important property of ARS is its invariance to translation, together with the orientation shift caused by the action of rotation (more details in Section 3.2).
ARS has been formulated and can be successfully used to perform rotation estimation, with versions that cover both the 2D (call it *ARS2D*) and 3D (*ARS3D*) cases. To complete the pose estimation pipeline, a simple translation estimation method able to obtain global optimality has been formulated.

3.1 Angular Radon Spectrum

A common representation used for sensory data across multiple sensors is the point cloud. Consider a point cloud where points are represented only through their 3D coordinates in space. The ARS rotation estimation algorithm takes a partially overlapping couple of source and destination *Gaussian mixture models* (GMMs) as input and returns the 3D rotation linking them. Kernel smoother is an established technique to provide continuous, smooth and adaptable estimation of a function from sample data. In particular, GMMs can effectively approximate the point density using kernels

with unimodal pattern. Although GMM is defined a probability density function, in this context such model is just a representation for the density of points sampled from the surface of obstacles. Scientific literature includes examples of a similar utilization of kernel models, i.e. normal distribution transform [100, 4]. This means that a relevant problem for using ARS on point clouds is how to "convert" the sensory output point clouds into input GMMs for ARS. This conversion step will be investigated more in-depth in Section 3.3.

From this point on, only GMMs are considered. One important classification of GMMs separates isotropic GMMs from the anisotropic GMMs. By definition, an isotropic GMM is composed only of isotropic Gaussians i.e., with covariance matrix $\mathbf{\Sigma} = \sigma I$. Under the isotropy assumption, ARS has a known closed-form [19]. The anisotropic case is analyzed more in-depth in Section 4.1.

3.1.1 Radon Transform of d-dimensional Gaussian Mixtures

We start from the Gaussian mixture model (GMM) representing a point cloud

$$f(\mathbf{r}) = \sum_{i=1}^{n_p} w_i \, f_i(\mathbf{r}) = \sum_{i=1}^{n_p} w_i \, \mathrm{n}\,(\mathbf{r} - \boldsymbol{\mu}_i, \mathbf{\Sigma}_i) \tag{3.1}$$

where in our case $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and $\mathbf{\Sigma}_i \in \mathbb{R}^{d\times d}$. When reasoning in the space the Radon Transform (RT) is computed on a plane. The parametric equation of the $(d-1)$-dimensional hyperplane is straightforward when using an orthonormal vector basis $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_d] \in SO(d)$ s.t.

$$\mathbf{r} = \mathbf{U}\,\mathbf{t} \tag{3.2}$$

where $\mathbf{t} = [t_1, ..., t_d]^\top$ is the vector of coefficients of linear combination of the $\mathbf{u}_i$ corresponding to point $\mathbf{r}$. Any hyperplane π is represented by fixing the normal vector $\mathbf{u}_1$ and the distance of π from the origin of reference frame $t_1 = \rho$. Hence, any point $\mathbf{r}$ in the hyperplane π is parameterized w.r.t. free variables $t_2, ..., t_d$

$$\mathbf{r}_\pi(t_2, ..., t_d) = \rho\mathbf{u}_1 + \sum_{i=2}^{d} t_d \mathbf{u}_d = \mathbf{U}_\perp \mathbf{t}_\perp + \mathbf{U}_\parallel \mathbf{t}_\parallel \tag{3.3}$$

Change of variables

$$\mathbf{U}_\perp = [\mathbf{u}_1],\ \mathbf{U}_\parallel = [\mathbf{u}_2, ..., \mathbf{u}_d]\ , \tag{3.4}$$

$$\mathbf{t}_\perp = t_1 \text{ and } \mathbf{t}_\parallel = [t_2, ..., t_d]^\top \tag{3.5}$$

to show that this choice of parameters can be generalized to hyperplanes of size $p < d$ in arbitrary spaces.

The RT of the GMM is computed as the integral of the sum of the Gaussian kernels over a generic hyperplane π. The integral over the hyperplane is the integral w.r.t. the parameters of the plane $\mathbf{t}_\parallel$

$$\mathscr{R}[f](\mathbf{U}_\perp, t_\perp) = \int_{\mathbb{R}^p} f(\mathbf{U}\mathbf{t})\ \mathbf{dt}_\parallel = \sum_{h=1}^{n_p} w_h \int_{\mathbb{R}^p} \mathrm{n}\left(\mathbf{U}\mathbf{t} - \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h\right)\ \mathbf{dt}_\parallel \tag{3.6}$$

The Gaussian kernels can be written in term of $\mathbf{t}$ instead of $\mathbf{r}$ after a change of variable. Now, it can be observed that the argument of the Gaussian distribution can be rewritten as

$$\begin{aligned}
(\mathbf{U}\mathbf{t} - \boldsymbol{\mu}_h)^\top\ \boldsymbol{\Sigma}_h^{-1}\ (\mathbf{U}\mathbf{t} - \boldsymbol{\mu}_h) &= (\mathbf{t} - \mathbf{U}^\top\boldsymbol{\mu}_h)^\top\ \mathbf{U}^\top\ \boldsymbol{\Sigma}_h^{-1}\ \mathbf{U}\ (\mathbf{t} - \mathbf{U}^\top\boldsymbol{\mu}_h) && (3.7)\\
&= (\mathbf{t} - \tilde{\boldsymbol{\mu}}_h)^\top\ (\mathbf{U}^\top\boldsymbol{\Sigma}_h\mathbf{U})^{-1}\ (\mathbf{t} - \tilde{\boldsymbol{\mu}}_h) && (3.8)\\
&= (\mathbf{t} - \tilde{\boldsymbol{\mu}}_h)^\top\ \tilde{\boldsymbol{\Sigma}}_h^{-1}\ (\mathbf{t} - \tilde{\boldsymbol{\mu}}_h) && (3.9)
\end{aligned}$$

where $\tilde{\boldsymbol{\mu}}_h = \mathbf{U}^\top\boldsymbol{\mu}_h$ and $\tilde{\boldsymbol{\Sigma}}_h = \mathbf{U}^\top\boldsymbol{\Sigma}_h\mathbf{U}$. The normalization constant of each Gaussian kernel can be adjusted accordingly (note that $\mathbf{U} \in SO(d)$ and $det(\mathbf{U}) = 1$). This is not surprising since the linear transformation of normally distributed random variables $\mathbf{r}$ in eq. (3.2) results into normal variables $\mathbf{t}$. Hence, the RT integral w.r.t. variable $\mathbf{t}_\parallel$ is equivalent to the marginalization of a Gaussian distribution

$$\begin{aligned}
\mathscr{R}[f_h](\mathbf{U}_\perp, t_\perp) &= \int_{\mathbb{R}^p} \mathrm{n}\left(\mathbf{t} - \tilde{\boldsymbol{\mu}}_h, \tilde{\boldsymbol{\Sigma}}_h\right)\ \mathbf{dt}_\parallel && (3.10)\\
&= \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h}\right) && (3.11)
\end{aligned}$$

where the marginal mean values and covariances are shown in the matrix block subdivision

$$\tilde{\boldsymbol{\mu}}_h = \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{\perp,h} \\ \tilde{\boldsymbol{\mu}}_{\parallel,h} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_\perp^\top \boldsymbol{\mu}_h \\ \mathbf{U}_\parallel^\top \boldsymbol{\mu}_h \end{bmatrix} \tag{3.12}$$

$$\tilde{\boldsymbol{\Sigma}}_h = \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h} & \tilde{\boldsymbol{\Sigma}}_{\perp\parallel,h} \\ \tilde{\boldsymbol{\mu}}_{\parallel\perp,h} & \tilde{\boldsymbol{\Sigma}}_{\parallel\parallel,h} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_\perp^\top \boldsymbol{\Sigma}_h \mathbf{U}_\perp & \mathbf{U}_\perp^\top \boldsymbol{\Sigma}_h \mathbf{U}_\parallel \\ \mathbf{U}_\parallel^\top \boldsymbol{\Sigma}_h \mathbf{U}_\perp & \mathbf{U}_\parallel^\top \boldsymbol{\Sigma}_h \mathbf{U}_\parallel \end{bmatrix} \tag{3.13}$$

In the following, the assumption that all the Gaussian kernels of the GMM are *isotropic* so that $\boldsymbol{\Sigma}_h = \sigma_h^2 \mathbf{I}$ is made. This assumption enables a strong simplification of mathematical evaluation. As a matter of fact, under such assumption $\tilde{\boldsymbol{\Sigma}}_h = \sigma_h^2 \mathbf{U}^\top \mathbf{U} = \sigma_h^2 \mathbf{I}$, and also the covariance of the marginal is reduced to a simple scalar $\tilde{\boldsymbol{\Sigma}}_{\perp\perp,h} = \sigma_h^2$.

3.1.2 Angular Radon Spectrum of d-dimensional Gaussian Mixtures

The *Angular Radon Spectrum* (ARS) is a functional defined as

$$\mathscr{S}[f](\mathbf{U}_\perp) = \int_{\mathbb{R}^{d-p}} \kappa(\mathscr{R}[f](\mathbf{U}_\perp, \mathbf{t}_\perp))\ \mathbf{dt}_\perp \tag{3.14}$$

where $\mathscr{R}[\cdot](\cdot)$ is the Radon Spectrum and the function $\kappa(x)$ is called *concentration function*. The concentration function needs to be superadditive (see [19]) and is usually $\kappa(x) = x^2$. The double product arising from applying the $\kappa(x) = x^2$ concentration function to the general Radon Spectrum equation can be simplified as follows:

$$\begin{aligned}
&\kappa(\mathscr{R}[f](\mathbf{U}_\perp, \mathbf{t}_\perp)) \\
&\quad = \left(\sum_{1 \leqslant h \leqslant n_p} w_h \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h}\right) \right)^2 \\
&\quad = \sum_{1 \leqslant h_1, h_2 \leqslant n_p} w_{h_1} w_{h_2} \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h_1}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1}\right) \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \\
&\quad = \sum_{1 \leqslant h_1, h_2 \leqslant n_p} w_{h_1} w_{h_2} \mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \cdot \\
&\qquad \cdot \mathrm{n}\left(\mathbf{t}_\perp - \bar{\boldsymbol{\mu}}_{h_1 h_2}, \bar{\boldsymbol{\Sigma}}_{h_1 h_2}\right)
\end{aligned} \tag{3.15}$$

where the formula of the product of two Gaussian probability density functions [101, sec. 8.1.8, p. 42] has been used to perform the following substitution

$$\bar{\boldsymbol{\mu}}_{h_1h_2} = \bar{\boldsymbol{\Sigma}}_{h_1h_2}(\tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_1}\tilde{\boldsymbol{\mu}}_{\perp,h_1} + \tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_2}\tilde{\boldsymbol{\mu}}_{\perp,h_2}) \tag{3.16}$$

$$\bar{\boldsymbol{\Sigma}}_{h_1h_2} = (\tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_2})^{-1} \tag{3.17}$$

Thus, the ARS integral in $\mathbf{t}_\perp$ of eq. (3.14) can be solved

$$\begin{aligned}\mathscr{S}[f](\mathbf{U}_\perp) &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\,\mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \\ &\quad \int_{\mathbb{R}^{d-p}} \mathrm{n}\left(\mathbf{t}_\perp - \bar{\boldsymbol{\mu}}_{h_1h_2}, \bar{\boldsymbol{\Sigma}}_{h_1h_2}\right)\,\mathbf{dt}_\perp \\ &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\,\mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right)\end{aligned} \tag{3.18}$$

where the integral of a Gaussian-like function over the whole domain is equal to 1 due to normalization.

The above derivation holds for general GMMs in a generic space with dimension d and Radon Transform for hyperplane of dimension $p < d$. Now, shift the focus specifically on the isotropic GMMs, making an assumption based on the definition mentioned in Section 3.1 and analyzed more in-depth in Section 3.3. In the isotropic case, the ARS yields

$$\mathscr{S}[f](\mathbf{U}_\perp) = \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\,\mathrm{n}\left((\boldsymbol{\mu}_{h_2} - \boldsymbol{\mu}_{h_1})^\top\mathbf{u}_1, \sigma^2_{h_1} + \sigma^2_{h_2}\right) \tag{3.19}$$

$$= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{e}^{-\frac{(\boldsymbol{\mu}^\top_{h_1h_2}\mathbf{u}_1)^2}{2(\sigma^2_{h_1}+\sigma^2_{h_2})}} \tag{3.20}$$

$$= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{e}^{-\frac{\mu^2_{h_1h_2}}{4(\sigma^2_{h_1}+\sigma^2_{h_2})}(1+\cos 2\gamma_{h_1h_2})} \tag{3.21}$$

where the brief notation $\boldsymbol{\mu}_{h_1h_2} = \boldsymbol{\mu}_{h_2} - \boldsymbol{\mu}_{h_1}$, $\mu_{h_1h_2} = \|\boldsymbol{\mu}_{h_1h_2}\|$ is adopted, and $\gamma_{h_1h_2}$ is the angle between $\boldsymbol{\mu}_{h_1h_2}$ and $\mathbf{u}_1$. Hence, there is an ARS kernel for any pair (h_1, h_2)

of GMM kernels, that are referred to using the notation

$$\mathscr{S}\left[f_{h_1h_2}\right] = \mathrm{e}^{-\lambda_{h_1h_2}(1+\cos 2\gamma_{h_1h_2})} \tag{3.22}$$

$$\lambda_{h_1h_2} = \frac{\mu_{h_1h_2}^2}{4(\sigma_{h_1}^2+\sigma_{h_2}^2)} \tag{3.23}$$

The equation of ARS kernel is identical to the one given in [19], if the Gaussian mixture kernels are also identical $\sigma_h = \sigma$ for all $h = 1,\ldots,n_p$. The quadratic complexity of ARS is apparent, as it describes "how much" aligned every couple of Gaussians is w.r.t. a pencil of parallel (hyper-)planes defined by the normal vector $U_\perp = u_1$. In this present form, ARS is hard to be utilized for its main purpose, which is the estimation of rotation. Instead, the ARS kernel $\mathscr{S}\left[f_{h_1h_2}\right]$ is more easily treatable in the frequency domain, at least for this specific goal. Because of this, it will be reformulated using the Fourier Series in 2D (see Section 4.1.2) and the Spherical Harmonics in 3D (see Sections 5.1, 5.2).

3.2 ARS Rotation-Shift and Translation Invariance

The properties of ARS presented in the following proposition are key to showing the potential of using it to evaluate pairwise rotations:

Proposition 1. *Let* $f(\mathbf{r}) = \sum_i w_i\ \mathrm{n}\left(\mathbf{r}-\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right)$ *be a GMM (as in eq.* (3.1)*), and let its ARS be* $\mathscr{S}\left[f(\mathbf{r})\right]\mathbf{U}_\perp$*), (after the base change in eqs.* (3.4)*,* (3.5)*). Let also* $\mathbf{v}$ *be a translation vector in* $\mathbb{R}^d$*. Then,* $\mathscr{S}\left[f(\mathbf{Rr}+\mathbf{v})\right](\mathbf{U}_\perp) = \mathscr{S}\left[f(\mathbf{r})\right](\mathbf{RU}_\perp)$.

Proof. Starting from what has been stated in eq. (3.18):

$$\mathscr{S}\left[f\right](\mathbf{U}_\perp) = \sum_{1\leqslant i,j\leqslant n_p} w_i w_j \mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,i}-\tilde{\boldsymbol{\mu}}_{\perp,j},\tilde{\boldsymbol{\Sigma}}_{\perp\perp,i}+\tilde{\boldsymbol{\Sigma}}_{\perp\perp,j}\right) \tag{3.24}$$

where

$$\tilde{\boldsymbol{\mu}}_{\perp,i}-\tilde{\boldsymbol{\mu}}_{\perp,j} = \mathbf{U}_\perp^\top(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j) \tag{3.25}$$

$$\tilde{\boldsymbol{\Sigma}}_{\perp,i}+\tilde{\boldsymbol{\Sigma}}_{\perp,j} = \mathbf{U}_\perp^\top\boldsymbol{\Sigma}_i\mathbf{U}_\perp+\mathbf{U}_\perp^\top\boldsymbol{\Sigma}_j\mathbf{U}_\perp = \mathbf{U}_\perp^\top(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)\mathbf{U}_\perp \tag{3.26}$$

which, in turn, come from subtracting a $h = j$-index term from an $h = i$-index term, in eqs. (3.12) and (3.13).

Now, consider applying a roto-translation to the argument of the GMM f:

$$f(\mathbf{Rr}+\mathbf{v}) = \tag{3.27}$$
$$\sum_i w_i \, \mathrm{n}\,(\mathbf{Rr}+\mathbf{v}-\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \tag{3.28}$$
$$\sum_i w_i \, \mathrm{n}\left(\mathbf{r}\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}), \mathbf{R}^\top\boldsymbol{\Sigma}_i\mathbf{R}\right) \tag{3.29}$$

which comes directly from observing that:

$$(\mathbf{Rr}+\mathbf{v}-\boldsymbol{\mu}_i)^\top\boldsymbol{\Sigma}^{-1}(\mathbf{Rr}+\mathbf{v}-\boldsymbol{\mu}_i) = \tag{3.30}$$
$$(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}))^\top\mathbf{R}^\top\boldsymbol{\Sigma}^{-1}\mathbf{R}(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v})) = \tag{3.31}$$
$$(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}))^\top(\mathbf{R}^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{R})(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v})) = \tag{3.32}$$
$$(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}))^\top(\mathbf{R}^\top\boldsymbol{\Sigma}\mathbf{R})^{-1}(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v})) \tag{3.33}$$

where in order to go from eq. (3.30) to (3.31), the following trick is used:

$$\mathbf{Rr}+\mathbf{v}-\boldsymbol{\mu}_i = \mathbf{R}(\mathbf{r}-\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v})) \ . \tag{3.34}$$

Looking at eqs. (3.30) to (3.33), the following equivalencies are apparent:

$$\boldsymbol{\mu}_i \to \mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}) \tag{3.35}$$
$$\boldsymbol{\Sigma}_i \to \mathbf{R}^\top\boldsymbol{\Sigma}_i\mathbf{R} \tag{3.36}$$

Now, consider the substitutions mentioned in eqs. (3.35) and (3.36), but for a roto-translated argument (i.e., $\mathbf{r} = \mathbf{Rr}+\mathbf{v}$), applying them to eqs. (3.25) and (3.26):

$$\tilde{\boldsymbol{\mu}}_{\perp,i} - \tilde{\boldsymbol{\mu}}_{\perp,j} = \mathbf{U}_\perp^\top(\mathbf{R}^\top(\boldsymbol{\mu}_i-\mathbf{v}) - \mathbf{R}^\top(\boldsymbol{\mu}_j-\mathbf{v})) = \tag{3.37}$$
$$\mathbf{U}_\perp^\top\mathbf{R}^\top(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j) = \mathbf{U}_\perp^\top\mathbf{R}^\top(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j) = (\mathbf{RU}_\perp)^\top(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j) \ ; \tag{3.38}$$

$$\tilde{\boldsymbol{\Sigma}}_{\perp,i} + \tilde{\boldsymbol{\Sigma}}_{\perp,j} = \mathbf{U}_\perp^\top\mathbf{R}^\top(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)\mathbf{RU}_\perp = (\mathbf{RU}_\perp)^\top(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)\mathbf{RU}_\perp \ . \tag{3.39}$$

To end the proof, compare eqs. (3.25), (3.38) and, respectively, eqs. (3.26), (3.39), it is apparent that the single Gaussians that compose the ARS spectra are as expected i.e., $\mathbf{U}_\perp$ becomes $\mathbf{RU}_\perp$ and translation $\mathbf{v}$ disappears. □

3.3 Point Clouds and GMMs

At this point, it is important to provide additional information on why and how ARS takes a GMM as input. An important definition regards the isotropy of a GMM has been introduced in previous sections. First, a Gaussian is said to be *isotropic* if its covariance matrix has the same eigenvalues and its distribution is identical from all the directions. GMMs that are composed only by isotropic Gaussian distributions are called isotropic themselves. Instead, GMMs that do not satisfy this property are called *anisotropic*. Of course, a very common representation for sensory data is the one of *point clouds*. The simplest way to convert a point cloud into a GMM is to associate each point of the cloud to an isotropic Gaussian with mean value μ on the point's coordinates, and arbitrary covariance matrix $\mathbf{\Sigma} = \sigma^2 I$. Normally, σ would vary depending on sensor uncertainty, and could even vary among points. The formulation of ARS in the isotropic case is leaner and can be taken as a good starting point. However, in general, the distribution of noisy point sets is more correctly and efficiently represented by general anisotropic Gaussian kernels. The analytical expression of ARS function allows full closed-form evaluation in the case of isotropic kernels. This differences are applicable to both the 2D case, as well as the 3D one.

3.4 Rotation Estimation

ARS can be seen as a stand-alone point cloud descriptor. However, as already said, the main goal behind its formulation is to use it for estimating the rotation that separates a couple of partially overlapping point clouds, call them source *(src)* and, respectively, destination *(dst)*. To perform this, the following rotation estimation pipeline based on ARS has been proposed.

1. simplification of the input source and destination GMMs $f_S(r)$ and $f_D(r)$ (only for Anisotropic ARS, see Section 4.1.1); the link between point clouds and GMMs has just been discussed in Section 3.3

2. computation of the ARS spectra of source and destination

3. expansion in frequency domain through Fourier Series (2D) or Spherical Harmonics (3D)

4. computation of the Fourier/Spherical Harmonics coefficients of the correlation function

5. optimization over the correlation function to obtain the rotation R^* for which the maximum is obtained

At this point of the presentation, it is useful to briefly describe the rotation estimation procedure in the simpler 2D case (the 3D case is tackled in Section 5.4) in order to in order to give a full view of the rotation estimation through ARS pipeline. Let the correlation function be $C[f_S, f_D]$. The computation of its global maximum through the lower and upper bounds of correlation on any angular interval $[\underline{\theta}, \overline{\theta}]$ are computed through simple interval arithmetic on sines and cosines of Fourier series. A *branch-and-bound* (*B&B*) procedure, has been proposed [19, 98] to solve the present planar case. The splitting of a generic interval $[\underline{\theta}^*, \overline{\theta}^*]$ in *B&B* iterations stops when the desired accuracy $\Delta\theta$ is reached, $\overline{\theta}^* - \underline{\theta}^* > \Delta\theta$. It can be empirically observed that low-frequency coefficients of correlation are predominant and $C[f_S, f_T](\delta)$ is a rather smooth function, since it is obtained as a convolution. Thus, its global maxima are well defined. The optimization procedure follows an almost identical procedure in the 2D anisotropic case (see Section 4.1.3) and in the GPU-enhanced isotropic case (see Section 4.4.2). Instead, the 3D case uses the same footprint, but with additional differences due to the increased complexity of the problem, as the rotation to be estimated presents 3 DOFs, one per each axis, compared to the only 1 DOF of the planar case. The 3D estimation procedure has been performed using Riemannian Optimization techniques, as will be detailed in Section 5.3.

3.5 Translation

To complete global registration, a branch-and-bound procedure inspired by [102] has been proposed and implemented as follows. The objective function to be maximized is the number of overlapping point pairs between destination and the translated source

point clouds. A point pair is overlapping if their distance is less than tolerance ε. Given a closed box $\mathscr{B} \subset \mathbb{R}^d$, the lower and upper bounds of the number of matching pairs are estimated. The lower bound is computed by counting the number of source points with a corresponding destination point belonging to the box $\mathscr{B}$ centered on the source point. The upper bound excludes from this counting the points clearly without matching. The translation is estimated as the center of the optimal box $\mathscr{B}_{opt}$ with largest number of inliers. The presented method is thought as an addition completing a more solid rotation estimation algorithm, in order to complete pose estimation, but it is still able to produce accurate results when combined with ARS, as it will be further discussed in Section 4.3. The proposed translation algorithm is able to work in d dimensions, so its adaptation from the 2D to the 3D case is seamless.

It has to be noted that the estimation of translation depends on the outcome of rotation. This enables the translation estimation algorithm to rely on a previous strong formulation that produces accurate rotation estimation results. Still, one problem that arises from the formulation of ARS is its periodicity. To be more explicit, in the planar case ARS estimates two values of rotation since it is π-periodic. Hence, the real rotation angle for a certain axis (also in 3D) is either δ^* or $\delta^* + \pi$. The assessment of translation enables disambiguation between the candidate values of rotation.

3.6 Discussion

The following Chapters 4 and 5 tackle the (planar) ARS2D and, respectively, the (spatial) ARS3D. The ARS2D chapter focuses on the anisotropic formulation and the GPU-enhanced parallelization of the isotropic case (Cud-ARS). The ARS3D chapter starts explaining the theoretical differences of applying ARS to the 3D case, compared to the 2D one. Both chapters then proceed illustrating the respective rotation estimation algorithms, and then present the experimental results for each of these subjects, comparing them against each other and against other state of the art algorithms.

Chapter 4

Rotation Estimation in 2D Domain Based on ARS

This chapter contains the description of the planar (2D) ARS formulation, with two major sections that cover anisotropic planar ARS, and a GPU-enhanced version of the isotropic version of ARS. The general aspects covered in the introduction to ARS (chapter 3) remain valid, but additional considerations for them to be put into practice in the case of 2D point clouds and GMMs are provided in the following. A relevant simplification comes from the fact that planar rotations have only 1 DOF instead of the 3 DOFs of the spatial case.

We start from the Radon Transform of a GMM, $\mathscr{R}[f](\mathbf{q}) = \sum_{i=1}^{n_p} w_i \int\limits_{\mathscr{F}_\mathbf{q}} \mathrm{n}\left(\mathbf{r} - \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\, d\mathbf{r}$. The marginal distribution $t_1 \sim \mathscr{N}(\tilde{\mu}_{i,1}, \tilde{\sigma}_{i,1}^2)$ has mean value and covariance equal to

$$\tilde{\mu}_{i,1} = \mathbf{u}_1^\top \boldsymbol{\mu}_i \quad \tilde{\sigma}_{i,1}^2 = \mathbf{u}_1^\top \boldsymbol{\Sigma}_i \mathbf{u}_1 \tag{4.1}$$

Thus, the RT of a single Gaussian kernel has the form of a Gaussian function as

$$\mathscr{R}[f_i](\theta, \rho) = \mathrm{n}\left(\rho - \tilde{\mu}_{i,1}, \tilde{\sigma}_{i,1}^2\right) \tag{4.2}$$

where we substituted $t_1 = \rho$ and the values of mean $\tilde{\mu}_{i,1}$ and variance $\tilde{\sigma}_{i,1}^2$ depend on the angle θ of $\mathbf{U}$.

In the 2D case, $\mathscr{F}_{\mathbf{q}}$ is a line represented by its polar parameters $\mathbf{q} = [\theta, \rho]^\top$ and the equation of a point in $\mathscr{F}_{\mathbf{q}}$ is

$$\mathbf{r}(\mathbf{t}) = t_1\ \mathbf{u}_1 + t_2\ \mathbf{u}_2 = \mathbf{U}\mathbf{t} \tag{4.3}$$

where $t_1 = \rho$ is a fixed constant, t_2 is the parameter associated to the points on the line, $\mathbf{u}_1 = \hat{\mathbf{u}}(\theta) = [\cos\theta, \sin\theta]^\top$ is the unitary vector orthogonal to the line and $\mathbf{u}_2 = \hat{\mathbf{u}}(\theta + \pi/2) = [-\sin\theta, \cos\theta]^\top$ corresponds to the line direction.

Hence, recalling eqs. (3.18) and (3.21), in the 2D case they become:

$$\mathscr{S}[f](\theta) = \int_{-\infty}^{+\infty} \kappa(\mathscr{R}[f](\theta,\rho))\ d\rho = \sum_{i=1}^{n_p}\sum_{j=1}^{n_p} w_i\ w_j\ \psi_{ij}(\theta) \quad \text{with} \tag{4.4}$$

$$\psi_{ij}(\theta) = \int_{-\infty}^{+\infty} \pi_{ij}(\rho,\theta)d\rho = \mathrm{n}\left(\tilde{\mu}_{i,1} - \tilde{\mu}_{j,1}, \tilde{\sigma}^2_{i,1} + \tilde{\sigma}^2_{j,1}\right) = \frac{1}{\sqrt{2\pi\ b_{ij}(\theta)}}\ \mathrm{e}^{-\frac{a_{ij}(\theta)}{2\ b_{ij}(\theta)}}$$

where π_{ij} is a double product of Gaussian densities, $\kappa(\cdot)$ is a superadditive function e.g., $\kappa(x) = x^2$, and

$$a_{ij}(\theta) = (\mathbf{u}_1^\top(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j))^2 = \frac{m_{ij}^2}{2}\ (1 + \cos(2\theta - 2\beta_{ij})) \tag{4.5}$$

$$b_{ij}(\theta) = \mathbf{u}_1^\top(\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)\mathbf{u}_1 = \hat{\sigma}_{ij}^2(1 + \delta_{ij}\cos(2\theta - 2\gamma_{ij})) \tag{4.6}$$

as derived in [98]. It is apparent that the ARS of a GMM has the form of a summation of kernels $\psi_{ij}(\theta)$ defined for each pair of Gaussians i, j. The cosine dependence of $a_{ij}(\theta), b_{ij}(\theta)$ allows expansion in Fourier series, which will be used during the optimization procedure for rotation estimation, as better detailed in Sections 4.1.2, 4.1.3.

Also, the general property of ARS shown in Section 3.2 applied to the 2D case has the following formulation. Let $f(\mathbf{r})$ be a GMM subject to rotation $\mathbf{R}(\delta) = \mathbf{R} \in SO(2)$ and translation $\mathbf{t} \in \mathbb{R}^2$. Then [19], its ARS is equal to

$$\mathscr{S}[f(\mathbf{R}(\delta)\ \mathbf{r} + \mathbf{t})](\theta) = \mathscr{S}[f(\mathbf{r})](\theta + \delta) \tag{4.7}$$

where the transformed spectrum does not depend on translation vector $\mathbf{t}$ and is shifted by angle δ.

4.1 Anisotropic ARS

In general, the distribution of noisy point sets is more correctly and efficiently represented by general anisotropic Gaussian kernels. This chapter begins by covering the point cloud-to-GMM conversion in Section 4.1.1 and then proceeds describing a general formulation of ARS (in the anisotropic case) in the planar case. Such generalization offers theoretical and practical advantages. First, anisotropic ARS better fits the distribution of sensor uncertainty using the right covariance matrix. Second, an arbitrary input GMM can be approximated and substituted by another GMM with a lower number of Gaussian kernels.

4.1.1 Simplification of Gaussian Mixture Models

The measurements provided by range sensors can be effectively modeled in the form of GMMs as previously discussed. A non-trivial step to be discussed regards how to accurately transfer the information contained in a point cloud into a Gaussian Mixture model. The anisotropic ARS be applied to 2D GMMs consisting of kernels with arbitrary covariance matrices. In the isotropic case, the GMMs are simply obtained from the sensory point clouds by initializing a 2D Gaussian $\mathrm{n}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ per each point $\mathbf{p} = [p_x, p_y]^\top$. The uncertainty on each point is given by its covariance matrix, whose value depends on the adopted noise model. In general, mean $\mu = p$ is set equal to the point's coordinates, and covariance matrix $\boldsymbol{\Sigma} = \sigma I$ is set equal to a scaled identity matrix, with variance σ determined on the basis of sensor's characteristics or on some other heuristics (in the simplest case, just let $\sigma = 1.0$ for all Gaussians). Under the isotropy assumption, the ARS has a known closed-form expression [19]. However, a different choice of the Gaussian kernels allows more accurate and compact description of the occupancy distribution. As a matter of fact, a GMM with anisotropic kernels can adapt better to the distribution using a reduced number of kernels. The anisotropic formulation of ARS allows a more accurate description of the structural elements of a point cloud e.g., points aligned or belonging to the same area or scene element can be grouped in a single non-isotropic Gaussian. The reduction in the number of Gaussians, although coming with an extra algorithmic step, enable a reduction

in the computational time of ARS. In order to perform this step, the following procedure for simplification and approximation of GMMs to significantly speed up the computation of ARS (quadratic in n_p) has been proposed.

Let $f^r(\mathbf{r})$ be the input GMM defined by parameters $\{(w_i^r, \boldsymbol{\mu}_i^r, \boldsymbol{\Sigma}_i^r)\}_{i=0,\dots,n_p^r}$ where w_i^r are the mixture weights, $\boldsymbol{\mu}_i^r$ the mean values and $\boldsymbol{\Sigma}_i^r$ the covariances. A significant application scenario occurs when the GMM consists of isotropic kernels, $\boldsymbol{\Sigma}_i^r = \sigma\mathbf{I}$. Simplification is achieved by substituting a subset of input Gaussians $\mathscr{I}_j$, called $f_{\mathscr{I}_j}^r(\mathbf{r})$, with the merged Gaussian $f_j^m(\mathbf{r})$, with parameters $(w_j^m, \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m)$, defined as

$$\begin{aligned} w_j^m &= \sum_{i\in\mathscr{I}_j} w_i^r \qquad \boldsymbol{\mu}_j^m = \frac{1}{w_j^m}\sum_{i\in\mathscr{I}_j} w_i^r\,\boldsymbol{\mu}_i^r \\ \boldsymbol{\Sigma}_j^m &= \sum_{i\in\mathscr{I}_j} \frac{w_i^r}{w_j^m}\left(\boldsymbol{\Sigma}_i^r + (\boldsymbol{\mu}_i^r - \boldsymbol{\mu}_j^m)(\boldsymbol{\mu}_i^r - \boldsymbol{\mu}_j^m)^\top\right) \end{aligned} \tag{4.8}$$

Here, the superscript ^{r} denotes the reference input GMM whereas the superscript ^{m} refers to the merged GMM. The pre-conditions for merging are that the input kernels in $\mathscr{I}_j$ lie inside the same region of space, and that the difference between the merged Gaussian j and input kernel, measured according to a proper metric, is less then a given threshold.

The *normalized integral squared error* (NISE) [103] is taken as distance for comparing two distributions over their domain. The NISE between the distributions $f_{\mathscr{I}_j}(\mathbf{r})$ and $f_j^m(\mathbf{r})$ is defined as

$$\text{nise}(f_j^m, f_{\mathscr{I}_j}^r) = \frac{D[f_{\mathscr{I}_j}^r - f_j^m]}{D[f_j^m] + D[f_{\mathscr{I}_j}^r]} \tag{4.9}$$

where the above integral norm $D[f]$ is defined as the integral of probability distribution f on the whole domain $\mathbb{R}^2$. In case $f_{\mathscr{I}_j}(\mathbf{r})$ is a sum of Gaussian kernels and $f_j^m(\mathbf{r})$ is a single Gaussian kernel, there are closed-form expressions for all the above

terms:

$$
\begin{aligned}
D[f^r_{\mathscr{I}_j} - f^m_j] &= \sum_{i \in \mathscr{I}_j} w^m_j w^r_i \, \mathrm{n}\left(\boldsymbol{\mu}^r_i - \boldsymbol{\mu}^m_j, \boldsymbol{\Sigma}^r_i + \boldsymbol{\Sigma}^m_j\right) \\
D[f^r_{\mathscr{I}_j}] &= \sum_{i_1, i_2 \in \mathscr{I}_j} w^r_{i_1} w^r_{i_2} \, \mathrm{n}\left(\boldsymbol{\mu}^r_{i_1} - \boldsymbol{\mu}^r_{i_2}, \boldsymbol{\Sigma}^r_{i_1} + \boldsymbol{\Sigma}^r_{i_2}\right) \\
D[f^m_j] &= (w^m_j)^2 \, \mathrm{n}\left(0, \boldsymbol{\Sigma}^m_j\right)
\end{aligned}
\tag{4.10}
$$

The NISE measures the difference in probability concentration, has closed-form expression for GMMs and is bounded between 0 and 1. A subset $\mathscr{I}_j$ of Gaussian kernels from f^r, all locally concentrated in the same region, is merged into a single Gaussian kernel f^m when their NISE is less than the threshold $nise_{thr}$.

The procedure to simplify the reference input GMM is presented by Algorithm 1. The general idea is to merge a subset of Gaussian kernels belonging to the same region into a merged Gaussian, to check the accuracy of such substitution using NISE, and to split the region into smaller subregions when the error is above a threshold. The kernels lying in the same regions are detected using an implicit *quadtree*, which is implemented as a list $\mathscr{T}$ of GMM kernels sorted according to Morton order [104]. Each mean vector $\boldsymbol{\mu}^r_i$ is encoded by its index vector $\mathbf{k}^r_i$ of the implicit grid of resolution q_{res} (line 7). List $\mathscr{T}$ is sorted in Morton order with respect to the keys $\boldsymbol{\mu}^r_i$ using Chan's *xor* trick [105]. To search the kernels lying in the same regions, we use an ordered list $\mathscr{T}$, which implicitly represents a *quadtree* data structure. The items are ordered according to their index vector $\mathbf{k}^r_i$ obtained by discretizing their mean vectors $\boldsymbol{\mu}^r_i$ with resolution step q_{res} (line 7).

In the main loop (lines 13-26), $\mathscr{T}$ is recursively split into smaller intervals until the simplified GMM is found. Each interval $[l,u[$ to be visited is extracted from queue $\mathscr{Q}$ and its level h in the quadtree (line 15) is computed as

$$
h = \max_{d=1,2} lev(k_{l,d}, k_{u,d}) \quad lev(a,b) = n_{bit} - nlz(a \oplus b) \tag{4.11}
$$

where n_{bit} is the number of bits of the integer type used for indices and $nlz()$ returns the number of leading zeros. Then, the algorithm assesses the Gaussian kernel f^m candidate for substituting the kernels $\mathscr{I}_j = [l,u[$ and evaluates the NISE (lines 17-18). If the NISE of the substitution is less than $nise_{thr}$ (and the tree level is large

Algorithm 1 SimplifyGMM

1: **function** SIMPLIFYGMM($f^r := \{(w_i^r, \boldsymbol{\mu}_i^r, \boldsymbol{\Sigma}_i^r)\}_{i=0,\ldots,n_p^r-1}$)
2: **Input**: GMM $f^r = \{(w_i^r, \boldsymbol{\mu}_i^r, \boldsymbol{\Sigma}_i^r)\}_{i=0,\ldots,n_p^r-1}$
3: **Output**: Anisotropic GMM $\mathscr{G}$, with parameters $(w_j^m, \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m)$
4: **Parameters**: $nise_{thr}$, q_{size}, q_{res};
5: $levelmax \leftarrow \lceil \log_2(q_{size}/q_{res}) \rceil$;
6: **for** $i = 0, \ldots, n_p^r - 1$ **do**
7: $\mathbf{k}_i^r \leftarrow \lfloor \boldsymbol{\mu}_i^r / q_{res} \rfloor$;
8: $\mathscr{T}.insert(\{\mathbf{k}_i^r, w_i^r, \boldsymbol{\mu}_i^r, \boldsymbol{\Sigma}_i^r\})$;
9: **end for**
10: sort keys $\mathbf{k}_i^r$ in $\mathscr{T}$ in Morton order [104];
11: $\mathscr{Q}.push([0, n_p^r[)$;
12: $\mathscr{G} \leftarrow \emptyset$;
13: **while** $\mathscr{Q} \neq \emptyset$ **do**
14: $[l, u[\leftarrow \mathscr{Q}.pop()$;
15: $h \leftarrow computeTreeLevel(l, u)$; ▷ eq. (4.11)
16: $\mathscr{I}_j \leftarrow \{i | l \leqslant i < u\}$;
17: compute $f_j^m := (w_j^m, \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m)$ from $\mathscr{I}_j$; ▷ eq. (4.8)
18: $nise_{cur} \leftarrow nise[f_j^m, f_{\mathscr{I}_j}^r]$; ▷ eq. (4.9)
19: **if** $h < levelmax$ and $nise_{cur} < nise_{thr}$ **then**
20: $\mathscr{G} \leftarrow \mathscr{G} \cup \{(w_j^m, \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m)\}$;
21: **else**
22: $m \leftarrow splitInterval(\mathscr{T}, l, u)$;
23: $\mathscr{Q}.push([l, m[)$;
24: $\mathscr{Q}.push([m, u[)$;
25: **end if**
26: **end while return** simplified GMM $\mathscr{G}$;
27: **end function**

enough), f^m is added to output Gaussian kernel set $\mathscr{G}$. Otherwise, the interval $[l, u[$ is split on its largest span dimension (lines 22-24). Hereinafter we assume that ARSs are computed on simplified GMMs $\mathscr{G}$ and omit the superscripts ^{r} and ^{m}. The substitution is accepted if the NISE is above a given threshold and the tree level h does not exceed threshold $levelmax$, which depends on maximum size of a quadtree quadrant q_{size}, to

avoid too coarse approximation.

4.1.2 Fourier Expansion and Pairwise Correlation of ARS

A quick reminder that the ARS of a GMM has the form of a summation of kernels $\psi_{ij}(\theta)$ defined for each pair of Gaussians i, j

$$\mathscr{S}[f](\theta) = \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} w_i \, w_j \, \psi_{ij}(\theta) \tag{4.12}$$

Then, the ARS in the form of Fourier series can be used to estimate the rotation between two GMMs representing the same scene observed from different viewpoints. Thanks to the invariance of ARS to translation and its rotation-shift stated in proposition 1, the computation of rotation angle is decoupled from translation. We only need a convenient metric to measure the similarity of the ARS of two input GMMs.

Let $f_S(\mathbf{r})$ and $f_T(\mathbf{r})$ be respectively the density functions of the source and the target point sets, with $\mathscr{S}[f_S](\theta)$ and $\mathscr{S}[f_T](\theta)$ as their corresponding GMM-ARSs. Suppose that $f_T(\mathbf{r})$ represents the transformed version of $f_S(\mathbf{r})$, except for noise and field-of-view issues commonly occurring in perception problems. The rotation between $f_S(\mathbf{r})$ and $f_T(\mathbf{r})$ can be found by searching the angular shift δ^* that maximizes the overlap between the two spectra. The overlap is measured by

$$C[f_S, f_T](\delta) = \frac{1}{\pi} \int_0^{\pi} \mathscr{S}[f_S](\theta + \delta) \, \mathscr{S}[f_T](\theta) \, d\theta \tag{4.13}$$

that represents the correlation of the two spectra. The best alignment δ^* is found when the similarity $C[f_S, f_T](\delta)$ between the two spectra is maximum.

The computation of correlation in eq. (4.13) is complex and not convenient if the source and destination ARSs are in the form of eq. (4.12). Indeed, each ARS consists of $O(n_p^2)$ kernels, one for each pair of Gaussians, and a kernel-by-kernel comparison would be computationally expensive. A better approach is to express each ARS kernel $\varphi_{ij}(\theta)$ as the sum of orthogonal function bases. Since the functions $\varphi_{ij}(\theta)$ are π-periodic, the natural choice is its Fourier series. Thus, the Fourier series of the ARS $\mathscr{S}[f](\theta)$ of the GMM is obtained by summing the corresponding Fourier coefficients of each φ_{ij}.

The closed formula of Fourier coefficients for isotropic GMMs [19] cannot be trivially extended to the anisotropic case. Hence, the Fourier series of each point pair is computed using *Fast Fourier Transform* (FFT) on n_f samples $\varphi_{ij}(\theta_k)$ taken at equally spaced sampling points $\theta_k = k\,\pi/n_f$ for $k = 0,\ldots,n_f-1$. Thus, the ARS of the GMMs in eq. (4.12) is approximated by the discrete Fourier

$$\mathscr{S}[f](\theta) \simeq a_0^f + \sum_{k=1}^{n_f} \left(a_k^f \cos(2k\theta) + b_k^f \sin(2k\theta) \right) \tag{4.14}$$

From the above expression of coefficients it is clear that the complexity of the estimation of $\{a_k^f, b_k^f\}_k$ is $O(n_p^2\ n_f \log n_f)$. It is quadratic in the number of points n_p, since there is an ARS kernel for each pair of input Gaussian kernels. It is linearithmic with respect to the Fourier order n_f, since the coefficients are computed using FFT. In practice, a relative low order $n_f = 64$ is sufficient for accurate representation.

4.1.3 Correlation and Maximum Optimization (Anisotropic ARS)

Then, the ARS $\mathscr{S}[f_S]$ and $\mathscr{S}[f_T]$ are expressed as Fourier series. We observe that the chosen correlation function $C[f_S, f_T]$ is a convolution of $\mathscr{S}[f_S]$ and $\mathscr{S}[f_T]$ and can also be represented as Fourier series. The formulas of the Fourier coefficients of $C[f_S, f_T]$, as well as the computation of its global maximum through a *branch-and-bound* (*B&B*) procedure, have been illustrated in previous work [19] and recapped in Section 3.4. The lower and upper bounds of correlation on any angular interval $[\underline{\theta}, \overline{\theta}]$ have been computed through simple interval arithmetic on sines and cosines of Fourier series. The splitting of a generic interval $[\underline{\theta}^*, \overline{\theta}^*]$ in *B&B* iterations stops when the desired accuracy $\Delta\theta$ is reached, $\overline{\theta}^* - \underline{\theta}^* > \Delta\theta$. We can empirically observe that low-frequency coefficients of correlation are predominant and $C[f_S, f_T](\delta)$ is a rather smooth function, since it is obtained as a convolution. Thus, its global maxima are well defined.

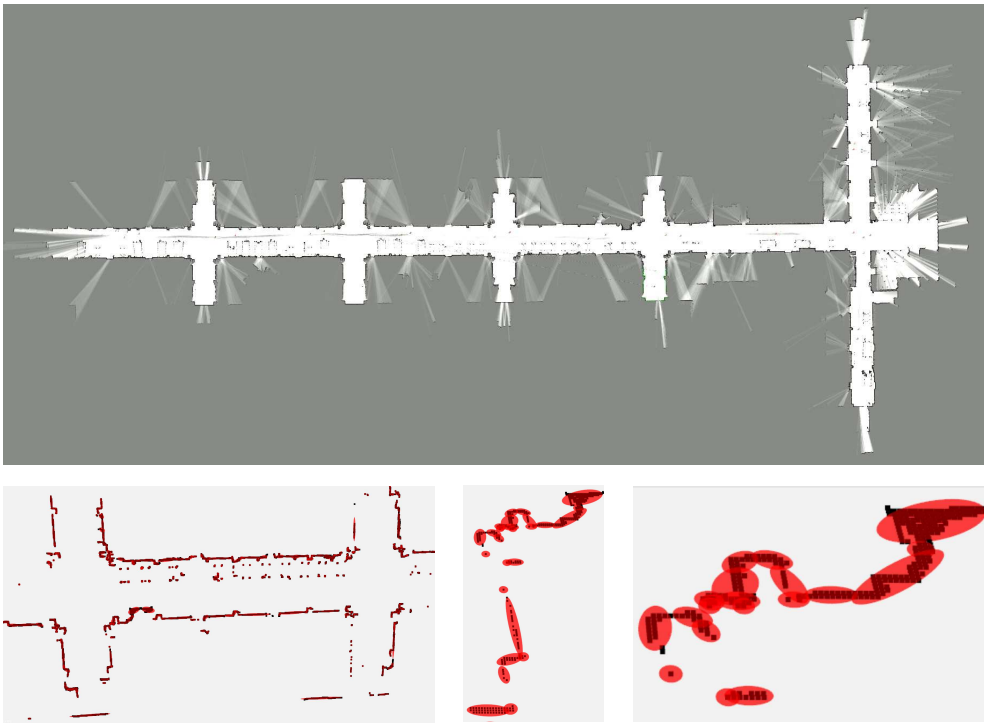


Figure 4.1: Anisotropic ARS GMM Simplification.
Above: maps dataset sample from Cartographer output.
Below: Gaussian simplification, zoomed at incremental levels of detail.

Table 4.1: Anisotropic ARS: Parameters configuration.

Description	Symbol	Value		
		mpeg7	maps	scans
ARS-Iso Fourier order	n_f		32	
ARS-Aniso Fourier order	n_f		64	
ARS standard deviation	σ_{min}	1.0	0.05	0.05
ARS tolerance on B&B	$\Delta\theta$		0.5°	
NISE threshold	$nise_{thr}$		0.15	
Quadtree resolution	q_{res}	1.0	0.05	0.05
Quadtree max quadrant size	q_{size}		$16 * q_{res}$	
HS angular resolution	$\Delta\theta_{hs}$	1.0	0.05	0.5
HS polar range resolution	$\Delta\rho_{hs}$	2.0	0.1	$0.05 \div 0.01$
HS polar range max	$\rho_{max,hs}$	400.0	400.0	150.0

4.2 Anisotropic ARS Experiments

The experiments presented in this section are designed to assess the performance on rotation estimation of the proposed anisotropic ARS (ARS-Aniso) and to compare it with other state-of-the-art algorithms. The methods used in the experiments are the original isotropic ARS (ARS-Iso) [19], HS [17], PCA (Principal Component Analysis) [106], a standard implementation of ICP, VFC [5] and TEASER [23]. In the case of isotropic ARS, the input points are used as mean values of an isotropic GMM. In the case of anisotropic ARS, there is more flexibility and the input GMM is simplified according to the procedure described in Section 4.1.1. The implementations of ARS-Aniso, ARS-Iso and HS are available in public repository[1]. ICP, VFC and TEASER are local search methods and their performance depends on initial guess of rotation, which is given by PCA. Both VFC and TEASER are able to remove most

[1] `https://github.com/dlr1516/ars`

wrong associations, but still rely on reasonable input estimation that is not guaranteed in our experiments. Moreover, TEASER is designed for 3D registration, not to deal with 2D point clouds. Three categories of datasets, discussed in the following subsections, have been used in the trials: the MPEG7 shape dataset, 3 occupancy map datasets, and 4 classic laser scan datasets. The parameters of the algorithms used in the experiments, in particular for isotropic and anisotropic ARS and HS, are reported in Table 4.1. Some of the parameters have different values depending on the dataset scale. Precision and accuracy of methods across the different types of dataset are analyzed in the following subsections. The MPEG7 tests are arguably more challenging and complete than the others, although they are run on a benchmark dataset, due to the large amount of situations in which ARS has to confront itself against the other methods. Still, the ones run on 2D scan datasets (and, in a more limited fashion, the ones on the occupancy grid maps) are the tests which bring the algorithm in the closest situation to a realistic use case.

Table 4.2 reports the number of comparison trials (i.e. the number of pair map), the average execution times and the number of GMM kernels processed by ARS-Aniso, ARS-Iso and HS. The experiments have been performed on processor Intel Core i9-11900F@2.50GHz with 32GB RAM. The average execution times of ARS-Aniso for datasets *mpeg7*, *backpack2d*, *unipr-dia*, *fr079*, *intel-lab* and *mit-csail* is less than those of ARS-Iso. The reduced time is straightforwardly attributed to the reduction to about 10% of the number of GMM kernels achieved by the procedure described in Section 4.1.1. When such reduction is not achieved (*backpack3d*) or is limited (*fr-clinic*), the flexibility of ARS-Aniso cannot be exploited. HS generally outperforms (sometimes only slightly) both ARS methods in terms of speed, but it requires parameter tuning and suffers from relative speed as discussed in Section 4.2.3.

4.2.1 MPEG7 Dataset

The original MPEG-7 database [107] consists of 1440 images of different shapes representing objects or animals. The contour points are used as noisy input points. The point set is translated and rotated according to known parameters and then distorted

Table 4.2: Anisotropic ARS: Execution Time Results

dataset	type	trial num	avg points num	ARS-Aniso avg kernel num	ARS-Aniso GMM kernel ratio	ARS-Aniso avg time [ms]	ARS-Iso avg time [ms]	HS avg time [ms]
mpeg7	images	1400	1444	30.93	8.99 %	129.81	376.29	2.08
backpack2d	maps	419	2140	177.19	8.28 %	13.30	625.02	5.82
backpack3d	maps	134	968	967.62	100.00 %	424.17	146.83	4.19
unipr-dia	maps	25	4100	340.74	8.31 %	54.49	2473.42	8.84
fr079	scans	1023	360	53.13	14.76 %	2.32	14.25	1.80 ÷ 14.13
fr-clinic	scans	377	180	136.43	75.69 %	13.93	3.51	1.63 ÷ 13.67
intel-lab	scans	905	180	39.61	22.01 %	1.47	3.66	1.46 ÷ 13.86
mit-csail	scans	760	361	53.53	14.83 %	2.60	14.23	1.81 ÷ 13.97

to assess the robustness in rotation estimation. The rotation angles are uniformly distributed on interval $[0, 180]$ *deg*, whereas the translation is uniformly distributed up to a maximum value approximately corresponding to the dimension of the object. Since this dataset has been used in our previous work [19], we briefly recapitulate the three kinds of distortion applied to the original point sets.

1. *Noise*. A Gaussian noise with standard deviation σ_{noise} in interval $0 \div 50$ is applied to each point coordinate.

2. *Occlusion*. All the points inside a circle centered in a randomly selected point of the set and with radius equal to a portion β of the dimension of the shape are removed ($\beta \in 0 \div 50\%$).

3. *Random Points*. The input set of n_{in} points is augmented by $\gamma\, n_{in}$ random points ($\gamma \in 0 \div 300\%$) drawn from a uniform distribution over the shape.

Figures 4.2(a)-(c) illustrate the results achieved using the discussed algorithms. The mean angular error is computed only on positives, i.e. with error less than 5°. Effective local registration algorithms like VFC perform well only on the noise distortion experiments, where the contour outline is preserved and point-to-point association is feasible. The poor performance of state-of-the-art TEASER may be attributed to inaccuracy of correspondences, to ineffectual tuning of parameters and to issues related to the application of a 3D registration algorithm to 2D domain. Most trials of TEASER terminate with the validity flag unset due to inconsistency in graph-based

inference. More investigation is required to understand the problem. In all other cases, the global correspondence-less methods ARS-Aniso, ARS-Iso and HS outperform the other techniques. In particular, they have similar results in occlusion tests since they are based on collinearity of points. The error of ARS-Aniso is usually close and intermediate between those achieved by ARS-Iso and HS.

4.2.2 Occupancy Grid Map Datasets

The second group of tests has been performed on GMMs generated from occupancy grid maps. The three datasets are *backpack2d* and *backpack3d* from the collection of Deutsches Museum, and *unipr-dia* acquired in the main hallways of the Dipartimento di Ingegneria e Architettura of the University of Parma. The raw laser scans and odometry measurements from these datasets have been processed by mapping tool *Carthographer* [108]. The maps built by Carthographer consist of several local occupancy grid submaps, which are used for rotation estimation. The cell centers of each occupancy grid submap are used as the mean values of an associated GMM.

The rotation estimation experiments are performed as follows. We select the candidate pairs of maps to be compared based on their index in the general map, which the Carthographer generally assigns according to the trajectory traveled by the robot. The majority of the maps with consecutive index are partially overlapping. The Carthographer tends to initialize the submap reference frames with orientation close to the origin frame. Thus, we manually added a random rotation to all the maps for testing arbitrary orientations.

The performances on rotation estimation of the compared algorithms are summarized in Figure 4.3. The first histogram represents the number of negative trials, i.e. with estimation error greater than 3°. Several negatives are due to non-overlapping submaps, even though with consecutive indices. The average rotation errors of ARS-Aniso, ARS-Iso and HS are close to 0.2° and are significantly lower than those achieved by local methods. ARS-Aniso performance is comparable to ARS-Iso and HS with no clear predominance of one method over the others.

4.2.3 Laser Scan Datasets

The third group of tests has been conducted on standard benchmarks of laser scans for robot localization and mapping applications: *fr079*, *fr-clinic*, *intel-lab* and *mit-csail*. Each of said datasets contains about 5000 scans. The goal of the experiments that have been run is to correctly estimate the rotation between subsequent scans in each dataset, and comparing the results obtained with the already mentioned six methods through the ground truth information contained in the datasets. Results are reported in Figure 4.4. Only subsequent scans with reciprocal ground truth rotations of at least 3° have been considered in order not to overrate the algorithm performance when limited rotation occurs. As for the Occupancy Grid Map experiments, several negative estimations are due to non-overlapping scans. The better results are achieved by ARS-Iso, ARS-Aniso and HS. For the latter dataset, an additional set of tests has been performed, in order to further assess the comparison between the mean execution times of ARS-Aniso and HS. By varying the HS polar range resolution from 0.05 to 0.01 (see corresponding line in table 4.1), the mean execution time of HS, initially lower the ones of both ARSs, saw a noticeable increase of up to 10 times.

4.3 Registration and Mapping Experiments

To attest the goal of building a SLAM pipeline based on ARS, a series of registration and mapping tests on navigation datasets has been performed. In general, ARS registration based on simple scan-to-scan alignment has been able to keep track of the trajectories, even if working more similarly to a scan matcher rather than a classic registration method (i.e., without proper loop closure or place recognition). To attest this, the proposed registration pipeline has been tested using a dataset acquired by a Pioneer 3DX robot in the main hallway of the Dipartimento di Ingegneria e Architettura of the University of Parma, which consists of a more than 100 meters long corridor with branches and tables. The measurements included in the dataset are the robot odometry, the laser scans collected by Sick LMS100, and the point clouds collected by multilayer LIDAR Velodyne VLP16. In particular, two sequences called

uniprdia_0 and *uniprdia_1* have been used.

The proposed ARS registration algorithm has been compared with Cartographer [108] and Hector SLAM (briefly, Hector) [109]. At each iteration, the estimation given by ARS is based only on the alignment of current laser scan with the previous one, without initial guess. Conversely, Hector and Cartographer are full mapping systems that align and merge each new scan with the current map. Moreover, Hector uses the initial guess provided by the robot odometry and Cartographer also integrates the 3D measurements from the LIDAR. The goal of such comparison is to display the robustness of ARS estimation, albeit based on scan-to-scan comparison.

Figures 4.10 and 4.11 display the occupancy grid maps, as well as the estimated trajectories obtained with the three methods. The occupancy grid maps are obtained through online collection and merging of the aligned laser scans using Octomap [110], without removing inconsistencies for a fair comparison of the three approaches. Even though it uses less data, ARS is able to estimate locally accurate trajectories. As it can be seen though, in dataset *uniprdia_0*, after the robot performs a U-turn, the algorithm loses track of the real orientation. The loss in orientation might be due to a non sufficient rate of consecutive scan matching during the turn, or even just a simple bad scan. More accurate are the results obtained in dataset *uniprdia_1*, where our method has been able to keep track of a more complex trajectory. It is well known that the absence of any kind of memory of previous states and maps while performing registration can lead to effects like this. However, these tests are here to show the stability of ARS' scan-to-scan rotation and translation estimation, even if the need for adding a more complex mapping pipeline to the ARS project still appears necessary.

Table 4.4 reports the Average Translational Error (ATE) and Average Rotational Error (ARE) for Hector and ARS with respect to the trajectory of Cartographer (used as ground truth) in the two sequences. As expected, ARS errors are larger, but significantly limited for a scan matching algorithm.

Another set of experiments has been conducted on standard benchmarks of planar laser scans for robot localization and mapping applications: *fr079*, *fr-clinic*, *intel-lab* and *mit-csail*. Each of said datasets contains about 5000 scans. The goal of the

conducted experiments is to correctly estimate the rigid transformation between subsequent scans in each dataset. Pose estimation tests have been separated into rotation and translation estimation respectively. For rotation, the results obtained through ARS3D have been compared against six methods through the ground truth information contained in the datasets, as explained in 4.2. Translation has been estimated after rotating the point clouds by an angle estimated through Isotropic ARS. Results are reported in Figure 4.5. ARS methods achieve an error on-par or lower than the other rotation estimation methods, while just a bit over the 1 cm scan resolution parameter when estimating translations. Only subsequent scans with reciprocal ground truth rotations of at least 5° have been considered in order not to overrate the algorithm performance when limited rotation occurs. It has to be noted that several failed estimations are due to non-overlapping scans.

4.4 Cud-ARS

Cud-ARS is a parallel algorithm for the registration of planar point clouds using ARS, implemented for execution on Nvidia GPUs through CUDA [99]. Cud-ARS has been proposed to enhance the planar isotropic case. ARS descriptors are represented by Fourier coefficients and depended on each point pair. The pairwise assessment of coefficients has been decomposed into independent tasks and assigned to different GPU cores. More specifically, the grid-like structure originates from splitting data into blocks. Computations are then performed among in-block point pairs by parallel threads, then by inter-block comparisons, and finally by a follow-up summation of partial results. In order to limit the computational load on each CUDA thread while also coping with the limited memory capabilities on each said thread, a matrix-like structure, adaptive to the size of the problem to be dealt with, has been implemented and is discussed in this dissertation. Our self-contained implementation also limits unnecessary dependencies and improves reusability for other projects and frameworks. Cud-ARS has been integrated into a full registration pipeline including translation estimation to perform pairwise scan alignment. Our experiments show a significant reduction in execution time guaranteed by GPU-based assessments

of rotation. Moreover, the mapping experiments on standard datasets show that the performance of Cud-ARS pairwise registration is comparable with tools performing state-of-the-art scan-to-map registration. As previously stated, the ARS computational complexity is dominated by the evaluation of a spectrum kernel for each possible pair of points, as it is clear from Equation (4.12). Since $\psi_{ij}(\theta) = \psi_{ji}(\theta)$, the final spectrum does not depend on the processing order of the points with indices i and j. Conventionally, the point with an index (say, i) in the external loop is called *source point*, and the one with an index (say, j) in the nested loop is the *destination point*. The simple idea behind the GPU enhancement of isotropic ARS is to execute the largest number of computations in parallel.

In order to maximize computational throughput across the GPU kernels and to distribute computation in an efficient manner by doing so, a virtual grid-like structure to compute ARS spectra has been introduced. The goal has been to keep the number of threads a power of 2, starting from a minimum of 32. To preserve the square shape of the grid, its last cells still perform computation on padding data as part of ARS spectra computation pipeline, even though the padding data are not related to real pairs of points.

Parameter *max_chunk_size* corresponds to the maximum number of points taken as input into one Cud-ARS processing CUDA grid. If the size of the input source or destination data is greater than *max_chunk_size*, the Cud-ARS coefficient update is iterated until all the source-to-destination point comparisons are processed. This step is necessary as the internal memory of modern GPUs is rather large, but still finite. An additional check is performed to avoid Cud-ARS coefficient computation steps with small data chunks. When chunks of data slightly surpass the maximum size allowed, but they still fit the GPU memory, they are processed in one step to avoid the additional iteration that would slow down the whole pipeline (especially as less CPU-GPU transfers with high throughput are more efficient than multiple transfers with less data).

4.4.1 Parallel Implementation

Cud-ARS is implemented as a three-step procedure. First, an indexed table of ARS coefficients is computed. The tid-th element of this table corresponds to the evaluation of ARS on points with indices i, j, with i, j computed as explained in the $getIJfromTID()$ method presented in Algorithm 2.

Algorithm 2 Cud-ARS: Obtain I and J from TID

Input: integer tid which stands for "total id": all the "strictly-triangular" couples of points in a cloud with n points
Outputs: integers i, j i.e., row-column indices in the parallelization grid
$nId \leftarrow n-1$; //indices vary between 0 and n−1
$tid\_tmp \leftarrow tid$
while $tid\_tmp >= 0$ **do** //find i
    $tid\_tmp -= (nId - i)$;
    $i \leftarrow i+1$;
end while
$i\_out \leftarrow i-1$;
$nId \leftarrow n-1$; // re-init
$tid\_tmp \leftarrow tid$
while $i > 0$ **do** //find j
    $tid\_tmp -= (nId - i)$;
    $i \leftarrow i-1$;
end while
$j\_out \leftarrow tid+1$;
return $i = i\_out, j = j\_out$

The tid indexing has been introduced in order to avoid excessive memory usage for storing useless computation outputs. As a matter of fact, the ARS coefficient matrix stores only the evaluation of ARS between points corresponding to non-null elements of the *strictly triangular* cost/matching matrix of the two datasets. Outputs i and j of Algorithm 2 correspond to the couple of point indices from source and,

respectively, destination sets to be processed.

The most significant and computationally expensive step of this first part of the algorithm is the ARS kernel computation, which is on its own composed of two steps: an evaluation of the *PNEBI* (Product of Negative Exponential and Bessel functions on the first kind) which is defined as

$$PNEBI(k,x) = 2\,\mathrm{e}^{-x}\mathscr{I}_k(x) \quad , \tag{4.15}$$

where $\mathscr{I}_k(x)$ is the modified Bessel function of the first kind of order k. The evaluation of $\mathscr{I}_k(x)$ is based on recurrence. Hence, it is convenient to evaluate all coefficients for $k = 0,\ldots,arsOrder$ and to store them into vector *pnebis*. Said vector is used to update the coefficient matrix, as illustrated in the discussion of Algorithm 3.

Algorithm 3 ARS Coefficient Downward Update: Parallelized in Cud-ARS

Input: *means* point set with $rowIdx, colIdx$ matrix indexing

Output: ARS coefficients **coeffs** vector for the input point set

Params: $fourierOrder, \sigma, pnebis$

For all $rowIdx$ **do**:

$factor \leftarrow weight = \frac{1.0}{(numPts^2) * \sqrt{(4\pi\sigma^2)}}$

$firstIdx = rowIdx * ncols$

$\textbf{coeffs}[firstIdx] \mathrel{+}= \frac{factor}{2} pnebis_0$

$\textbf{delta} \leftarrow means_j - means_i$

$\phi \leftarrow atan2(\textbf{delta}.y, \textbf{delta}.x)$

$cth2 \leftarrow cos(2\phi)$

$sth2 \leftarrow sin(2\phi)$

$sgn \leftarrow -1.0$

$cth \leftarrow cth2$

$sth \leftarrow sth2$

for $k = 1 : fourierOrder$ **do**

$evenIdx \leftarrow rowIdx * ncols + 2k$

$oddIdx \leftarrow rowIdx * ncols + (2k+1)$

$\textbf{coeffs}[evenIdx] \mathrel{+}= factor * pnebis[k] * sgn * cth$

$\textbf{coeffs}[oddIdx] \mathrel{+}= factor * pnebis[k] * sgn * sth$

$sgn \leftarrow -sgn$

$ctmp \leftarrow cth2 * cth - sth2 * sth$

$stmp \leftarrow sth2 * cth + cth2 * sth$

$cth \leftarrow ctmp$

$sth \leftarrow stmp$

end for

It can be noted that a large part of the computational load is due to the estimation of such $20 \div 30$-sized vector for each pair of points to be evaluated with ARS. The aforementioned Algorithm 3 runs on the GPU in a for-stride loop guarded by the following instructions:

$$\begin{aligned} &index = blockIdx.x * blockDim.x + threadIdx.x \\ &stride = blockDim.x * gridDim.x \\ &tot = gridDim.x * blockDim.x \\ &for(tid = index;\ tid < tot;\ tid \mathrel{+}= stride) \end{aligned} \tag{4.16}$$

where *index* runs through a single block, while *stride* is supposedly the total number of threads in the grid. The goal is to fit as many coefficient computations as possible into one single grid.

Then, ARS computation for rotation estimation proceeds by summing the coefficients across each Fourier order, i.e., summing them along each (virtual) column of the coefficient matrix. However, due to the large number of rows to be summed for each column, it has appeared more profitable to subdivide this summing procedure into two steps: first, a partial column-wise sum of the coefficient matrix entries in chunks of consecutive rows (each having a fixed number of rows *part_sum_numrows*) is been computed; then, the sum of these partial sums is computed (still column-wise).

One last important consideration to be made is on how to approach large input datasets. The considerable amount of data needed for each thread of the kernels' computation quickly fills the central memory of the GPUs, which for *general purpose* personal PCs rarely goes over 25 GB. Going even further into the exploitation of the separability of ARS coefficient parallel computation, the natural way for presented Cud-ARS to solve problems with input point sets with a size over $\sim$5000 points is subdivision of the datasets in chunks, and then the processing of each of the chunks separately. The 4096 value reported in Table 4.3 has been chosen as the appropriate maximum chunk size by empirical testing. The value of 256 representing the number of threads in each block has been selected in a similar fashion. Since kernel parallelization parameters vary depending on the input dataset chunk size, when

the number of input points is greater than *chunk_max_size*, an iterative procedure resembling the subdivision into partial sums and total sums explained for computing ARS coefficients of each Fourier order is deployed. First, the parallelization parameters are updated according to input data chunk size. Then, the computation of ARS coefficients is summed for each combination of data chunks among source and destination point sets. Finally, it has to be noted that even during this process, the strictly triangular indexing for the ARS coefficient matrices is kept even when the need arises to evaluate ARS across multiple different chunk combinations, coming from source and destination point sets. A graphical explanation of the Cud-ARS processing steps is provided in Figure 4.6.

Table 4.3: Cud-ARS: Parameters configuration.

Description	**Symbol**	**Value**
ARS Fourier order	n_f	32
ARS stdev	σ_{min}	1.0 (mpeg7), 0.05 (maps)
ARS tolerance on B&B	$\Delta\theta$	0.5°
Coeff Matrix Rows	$nrows$	$\frac{num\_pts*(num\_pts-1)}{2}$
Coeff Matrix Cols	$ncols$	$2n_f+2$
Prlz Grid Size	$grid\_sz$	$nrows$
Number of Blocks	$blks$	$\lfloor \frac{gridTotalSize}{pp.blockSz} \rfloor + 1$
Number of Threads	$threads$	256
Coeff Matrix Tot Size	$cffs\_mat\_tot\_sz$	$grid\_sz * ncols$
Max Chunk Size	max_chunk_size	4096

4.4.2 Correlation and maximum optimization (Cud-ARS)

The GPU-enhanced ARS coefficients computation does not change the fact that ARS can be effectively used to the estimate the rotation separating two point clouds that represent the same scene from different viewpoints. Also, the decoupling of rotation evaluation from translation evaluation remains valid. As previously stated, when there

is a rotation with angle δ between a source and a target point set represented by density functions respectively $f_S(\mathbf{r})$ and $f_T(\mathbf{r})$, the spectrum $\mathscr{S}[f_S]$ is the shifted copy of the $\mathscr{S}[f_T]$. The shift angle can be computed by searching the maximum of correlation between the two functions

$$C[f_S, f_T](\delta) = \frac{1}{\pi} \int_0^{\pi} \mathscr{S}[f_S](\theta + \delta)\, \mathscr{S}[f_T](\theta)\, d\theta \ . \tag{4.17}$$

As the correlation function optimization was already the least computationally intensive part of the ARS rotation estimation pipeline, no big modifications have been introduced for performing this step in Cud-ARS. The ARS coefficients, computed in parallel on the GPU, are expressed as Fourier series. As already discussed for the CPU-only version of ARS (section 3.4) the correlation function is elegantly expressed in form of convolution. The global maximum δ^* of $C[f_S, f_T]$ can be efficiently found through a branch-and-bound procedure on angular domain [19, 98]. ARS is π-periodic also in the Cud-ARS formulation, so the real rotation angle is either δ^* or $\delta^* + \pi$. The assessment of translation through the procedure described in 3.5 helps with the disambiguation between the two candidate values of rotation.

4.5 Cud-ARS Experiments

In order to evaluate Cud-ARS, different sets of experiments have been conducted. The first goal has been to assess if Cud-ARS can show the same accuracy performances as ARS and other state-of-the-art methods on commonly used datasets. Second, the proposed methods for scan-matching based registration have been tested on real world robotic simulations, assessing its ability in trying to reconstruct the trajectory and usability in building an occupancy grid map of the robot's movements. An implementation of Cud-ARS is available in a public repository[2]. In a similar fashion to what has been discussed in Section 4.2 for the Anisotropic ARS tests, the MPEG7 benchmarks present the largest number of purposely challenging scenarios for the algorithm to succeed, while the occupancy grid maps and 2D laser scans bring the algorithm closer to a realistic use case.

[2] `https://github.com/ErnestF22/cudars/`

4.5.1 Cud-ARS Rotation Estimation

Cud-ARS evaluation has been performed on three datasets (namely *mpeg7*, *map* and *scan*) that were used also in previous works, as across them can be found all common characteristics present in robotic tasks that use a good variety of sensors. *Mpeg7* datasets are composed of images of more than 1000 different shapes, that have been sampled as point clouds, also adding noise, introducing occlusion to cover some areas of the cloud, and adding random points to simulate sensory measurement errors. The *map* dataset is composed of occupancy grid maps obtained using the Cartographer [108] ROS tool on laser scans acquired at the University of Parma, and on public Deutsche Museum dataset. The *scan* dataset is made of laser scans typically used by the SLAM community, named after the place of acquisition: *fr-log*, *fr079*, *intel-lab* and *mit-csail*. Experiments on all these common robotic datasets showed as expected the same results for pose estimation as isotropic ARS, but with a substantial improvement in terms of speed. In these tests, Cud-ARS is compared against two previous versions of ARS (CPU Isotropic and Anisoptropic), and the Hough Spectrum [17] from Censi *et al*. Results are shown in Figures 4.7, 4.8 and 4.9. The speed-up of newly introduced Cud-ARS is easily noticeable across all experiments, as well as the limited growth in execution time when other algorithms heavily slow down instead. One algorithm that performs similarly to Cud-ARS is Hough Spectrum (HS). While HS slightly outperforms Cud-ARS on some tests, we can still see that their speed is always very similar, and that they are constantly much faster than their counterparts. All this at no cost in terms of accuracy and precision, which stay as discussed in previous works. Furthermore, the variance in terms of mean execution time is much higher for previous versions of ARS, which means that the newly introduced Cud-ARS can see a great increase in the number of potential applications, as its performance is reasonably constant when dealing with diverse types of datasets (for example in terms of the number of input points) whose elaboration may have previously required an excessive amount of time for on-line processing.

Dataset	Length	Hector		ARS	
	[m]	ATE	ARE	ATE	ARE
		[%]	[10^{-2}°/m]	[%]	[10^{-2}°/m]
uniprdia_0	262.28	3.87	7.18	19.78	31.66
uniprdia_1	180.34	3.14	6.66	11.06	32.58

Table 4.4: Average Translational Error (ATE) and Average Rotational Error (ARE) obtained by Hector and ARS on the given sequences of datasets *uniprdia_0* and *uniprdia_1*.

4.6 Discussion

This chapter has presented the 2D version of ARS, focusing on the more general anisotropic formulation and on Cud-ARS, a parallel, GPU-based formulation for the isotropic case. The experimental results, comparing the proposed methods against each other, and against other state of the art algorithms, have been discussed and can be summed up as follows. Anisotropic ARS is able to achieve great reduction in the computational time required to register a pair of point clouds, with very limited change in terms of accuracy, especially when the preliminary GMM reduction procedure is able to consistently reduce the number of input Gaussians. Cud-ARS is able to obtain substantial improvements in terms of speed over its counterparts, by being able to execute its most computational heavy step (i.e., computation of source and destination spectra coefficients) in a highly parallelized fashion. Also, the capabilities of 2D ARS in the context of iterative registration and mapping through consecutive scan-alignment have been examined.

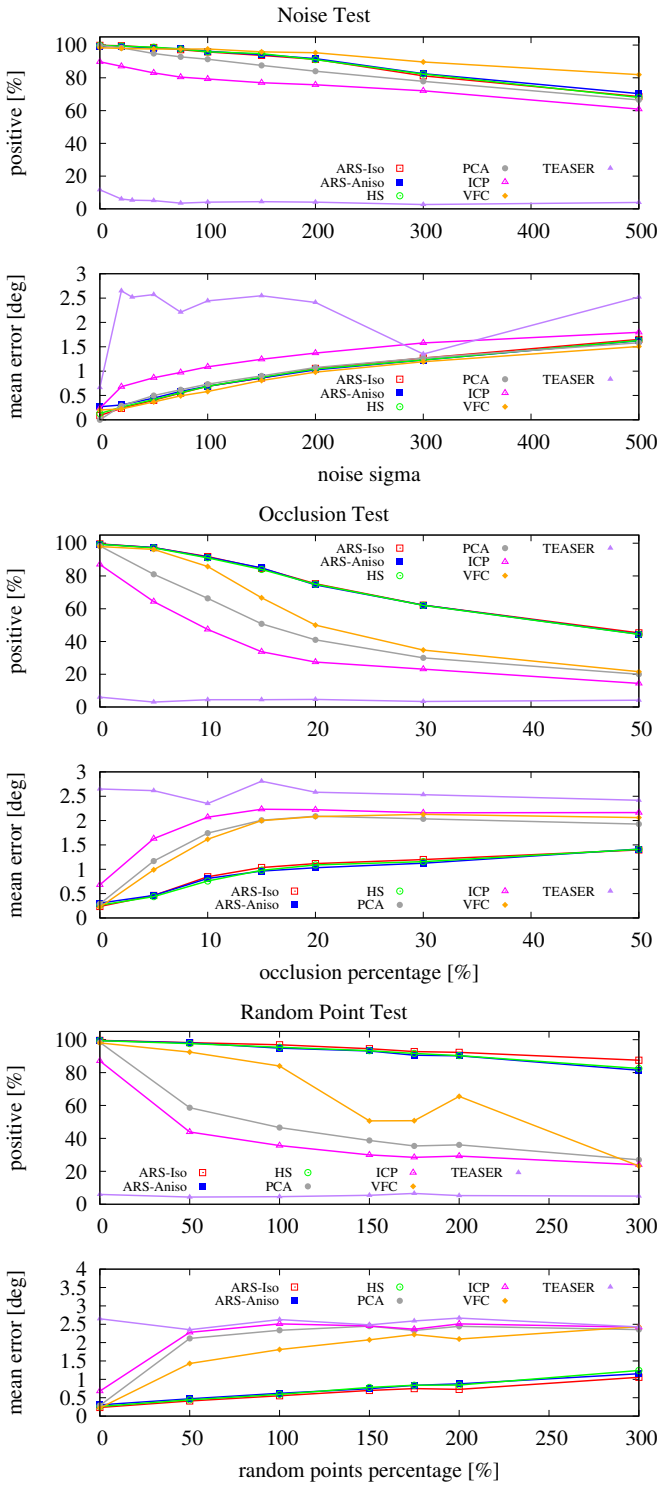


Figure 4.2: Anisotropic ARS: Positive estimation percentage (top) and mean angular error (bottom) obtained by ARS-Aniso, ARS-Iso, HS, PCA, ICP, VFC and TEASER on three experiments: (a) additive Gaussian *noise* with different standard deviation σ_{noise}, (b) *occlusion* with different occlusion rates β (in percentage), (c) *random points* with different rates γ (in percentage).

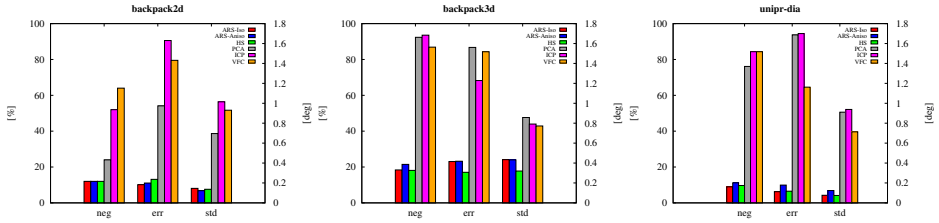


Figure 4.3: Anisotropic ARS: Negative estimation percentage, average rotation mean error [°] and standard deviation obtained in rotation accuracy tests on occupancy grid datasets *backpack2d*, *backpack3d* and *unipr-dia* (from left to right).

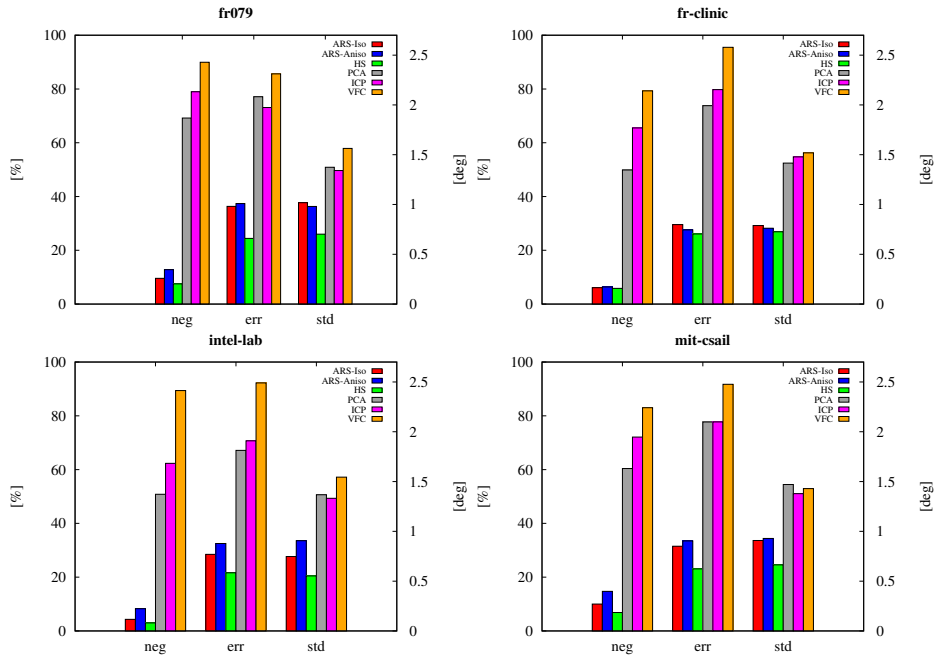


Figure 4.4: Anisotropic ARS: Negative estimation percentage, average rotation mean error [°] and standard deviation obtained in rotation accuracy tests on scan datasets *fr079*, *fr-clinic*, *intel-lab* and *mit-csail* (from left to right, top to bottom).

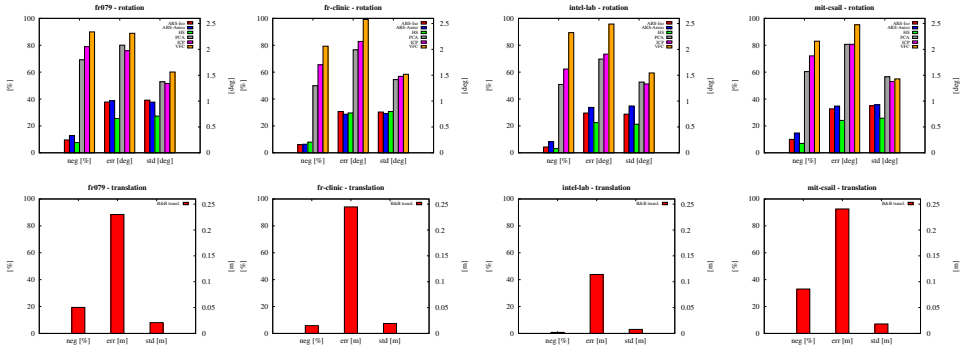


Figure 4.5: Anisotropic ARS: Registration accuracy in estimation of rotation (top) and of translation (bottom) on scan datasets *fr079*, *fr-clinic*, *intel-lab* and *mit-csail* (from left to right). For each set of tests, negative (failed) estimation percentage, average rotation error, standard deviation and translation mean error [$^\circ$, respectively $[m]$] are reported.

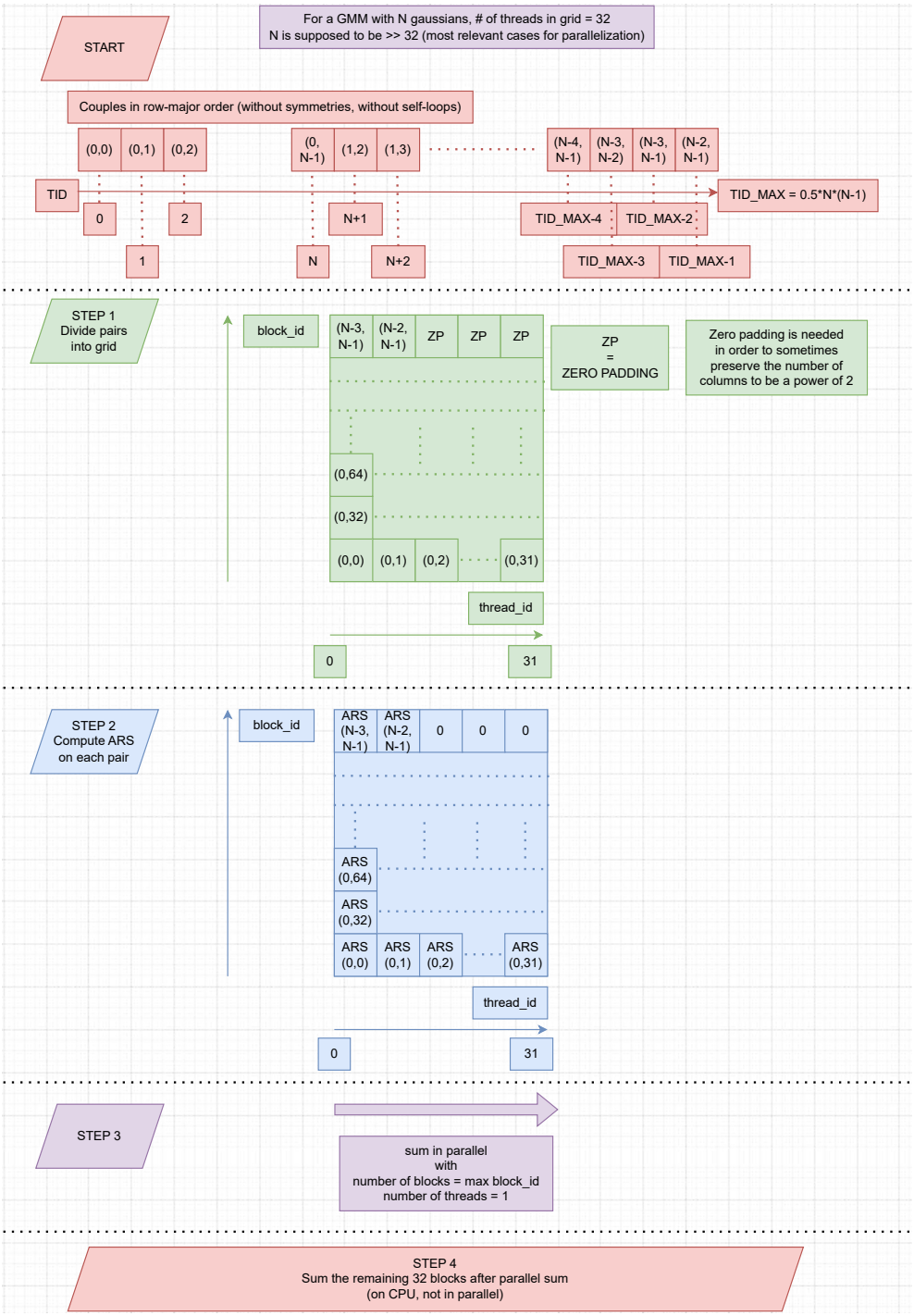


Figure 4.6: Cud-ARS graphical explanation

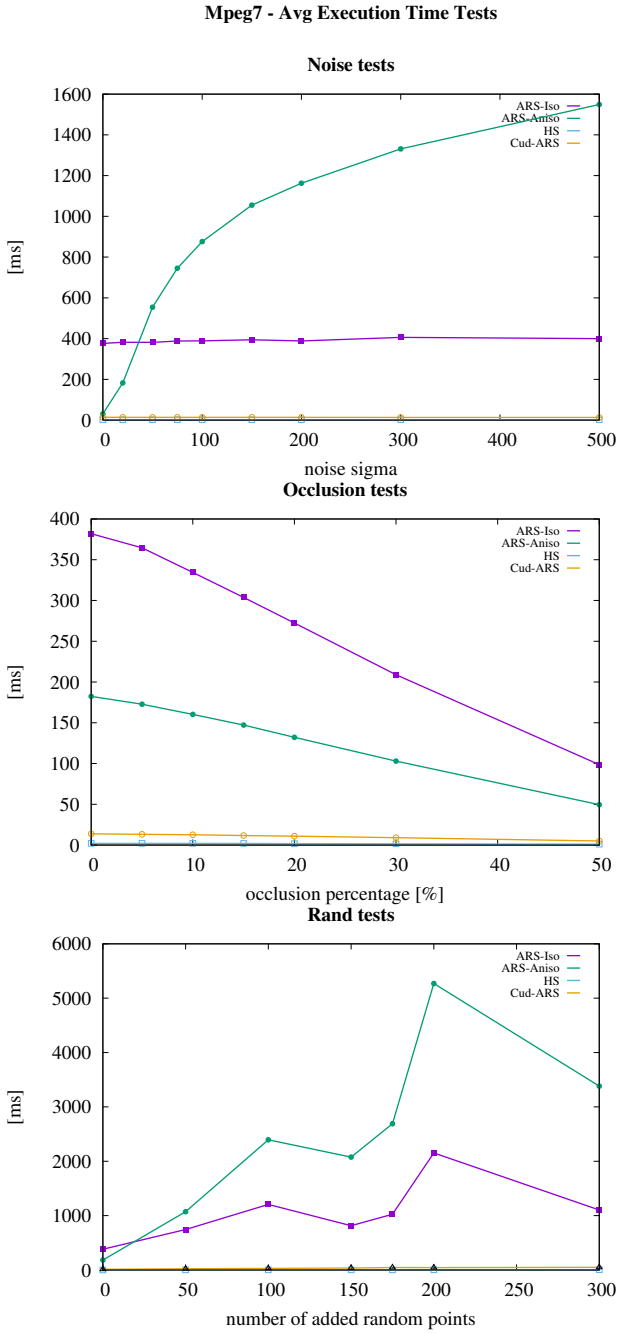


Figure 4.7: Cud-ARS: MPEG7 dataset execution times results

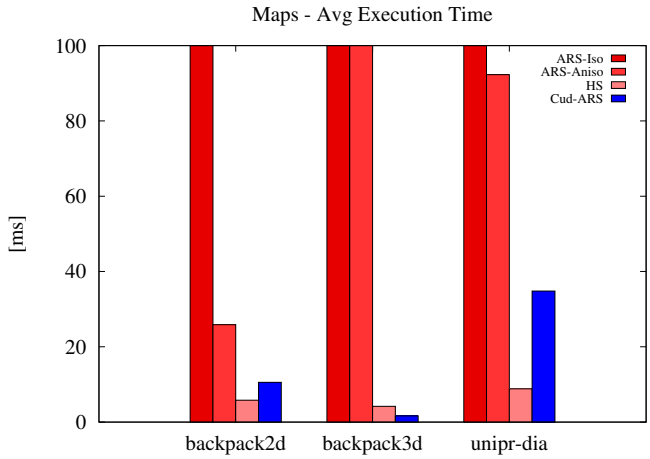


Figure 4.8: Cud-ARS: Maps dataset execution times results

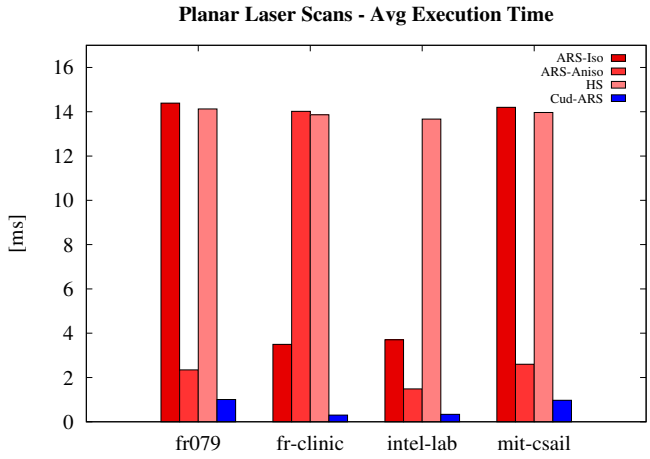
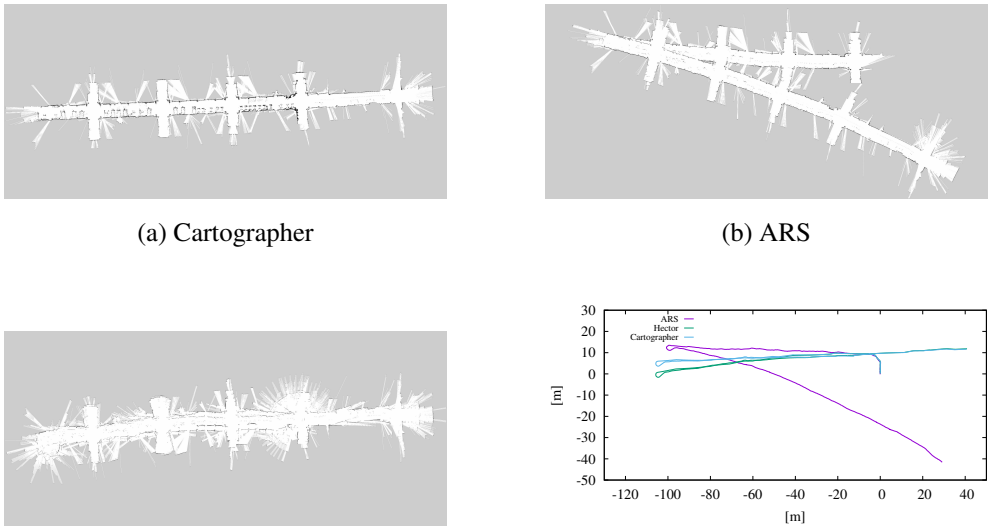


Figure 4.9: Cud-ARS: Scan dataset execution times results



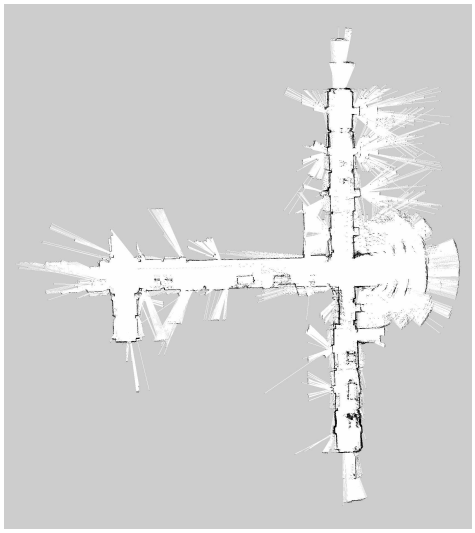
(a) Cartographer

(b) ARS

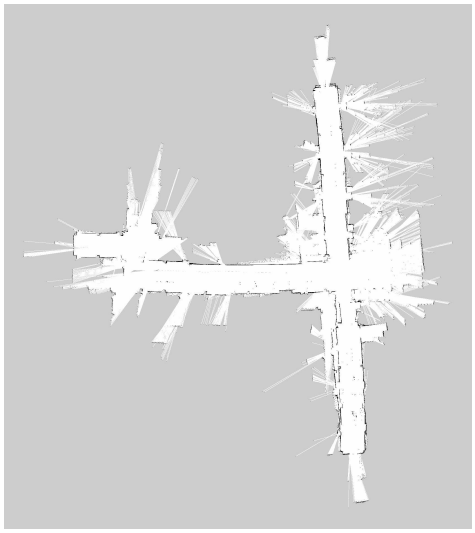
(c) Hector

(d) Trajectories

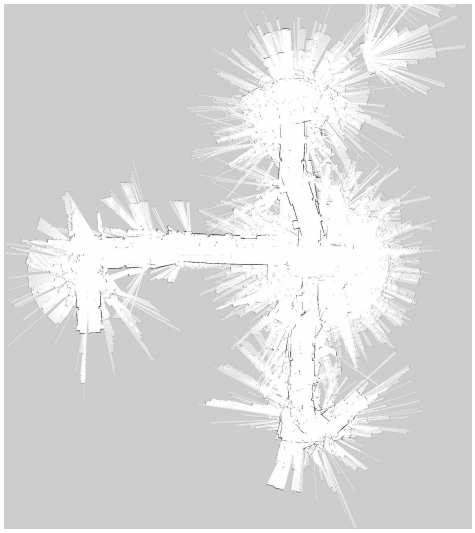
Figure 4.10: Occupancy grid maps and estimated trajectories of dataset *uniprdia_0* obtained using Cartographer, Hector SLAM and ARS-based registration. The occupancy grid maps are computed using Octomap that overlaps online raw laser scan data. Error propagation after the U-turn is visible.



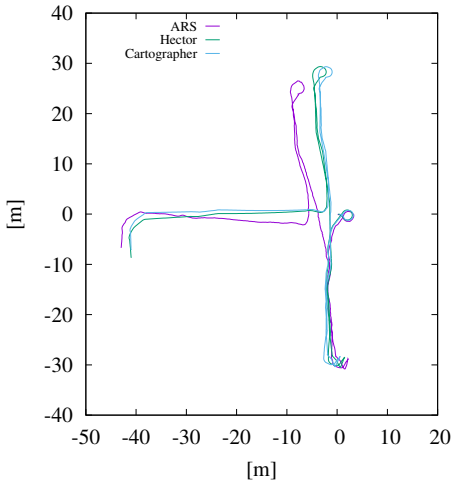
(a) Cartographer



(b) ARS



(c) Hector



(d) Trajectories

Figure 4.11: Occupancy grid maps and estimated trajectories of dataset *uniprdia_1* obtained using Cartographer, Hector SLAM and ARS-based registration. The occupancy grid maps are computed using Octomap that overlaps online raw laser scan data. Here, a more complex trajectory is kept track of with limited error.

Chapter 5

Rotation Estimation in 3D Domain Based on ARS

Now, let us focus on the tridimensional formulation of ARS, and on how to use it for estimating rotations in the space. In particular, this chapter presents the formulation of ARS for the 3D isotropic case. The equation of the RT for the specific 3D case in the conditions and notation discussed in Chapter 3 yields

$$\mathscr{R}[f](\theta,\varphi,\rho) = \sum_{h=1}^{n_p} w_h \, \mathrm{n}\left(\rho - \boldsymbol{\mu}_h^\top \mathbf{u}_1(\theta,\varphi), \sigma_h\right) \tag{5.1}$$

where $\mathbf{U}_\perp = \mathbf{u}_1(\theta,\varphi)$ is the normal unitary vector of the plane (parameterized with two angles θ and φ), $t_\perp = t_1 = \rho$ is the distance of the plane from the axis, and $\tilde{\boldsymbol{\Sigma}}_{\perp\perp,h} = \sigma_h$. In the presentation occasional references to the general formulation can still be found, but the adoption of the specific form in eq. (5.1) will help to derive a closed-form and to proceed with less abstract reasoning. It is important to point out the term $\boldsymbol{\mu}_h^\top \mathbf{u}_1(\theta,\varphi)$ which is a dot product. Unitary vector $\mathbf{u}_1$ can be parameterized in different ways as a point in 3D sphere, e.g.

$$\mathbf{u}_1(\theta,\varphi) = [\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta]^\top$$

but there are other possible choices. Also the mean vector can be written accordingly in polar coordinates

$$\boldsymbol{\mu}_h = \mu_h \left[\sin\theta_h \cos\varphi_h, \sin\theta_h \sin\varphi_h, \cos\theta_h\right]^\top$$

where $\mu_h = \|\boldsymbol{\mu}_h\|$ and

$$\begin{aligned} \boldsymbol{\mu}_h^\top \mathbf{u}_1 &= \mu_h(\cos\theta\cos\theta_h + \sin\theta\sin\theta_h\cos(\varphi-\varphi_h)) \\ &= \mu_h \cos\gamma_h \end{aligned} \tag{5.2}$$

The relation between $\cos\gamma_h$, where $\gamma_h = \angle\boldsymbol{\mu}_h\mathbf{u}_1$ is the angle between vectors $\boldsymbol{\mu}_h$ and $\mathbf{u}_1$, and the first dot product equation is the spherical cosine law. In this context, it is important to define the *spherical phase* as the angle between the variable vector $\mathbf{u}_1$ on unit sphere and other constant vector $\mathbf{a}$ (in this case $\mathbf{a} = \boldsymbol{\mu}_h$, but other vectors will be used in the following), i.e. $\angle\mathbf{a}\mathbf{u}_1$. Although it seems a trivial operation, the appearance of the cosine term will help the derivation of ARS kernel. The *Angular Radon Spectrum* (ARS) is defined as

$$\mathscr{S}[f](\mathbf{U}_\perp) = \int_{\mathbb{R}^{d-p}} \kappa\left(\mathscr{R}[f](\mathbf{U}_\perp, \mathbf{t}_\perp)\right)\, \mathbf{dt}_\perp \tag{5.3}$$

where the function $\kappa(x)$ is called *concentration function* and is usually $\kappa(x) = x^2$. Temporarily slipping back to the more general parameters instead of the simplified form of eq. (5.1) leads to the following expression

$$\begin{aligned} &\kappa\left(\mathscr{R}[f](\mathbf{U}_\perp, \mathbf{t}_\perp)\right) \\ &= \left(\sum_{1\leqslant h\leqslant n_p} w_h \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h}\right)\right)^2 \\ &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1} w_{h_2} \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h_1}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1}\right) \mathrm{n}\left(\mathbf{t}_\perp - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \\ &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1} w_{h_2} \mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \cdot \\ &\quad \cdot \mathrm{n}\left(\mathbf{t}_\perp - \bar{\boldsymbol{\mu}}_{h_1h_2}, \bar{\boldsymbol{\Sigma}}_{h_1h_2}\right) \end{aligned} \tag{5.4}$$

where the formula of the product of two Gaussian probability density functions [101, sec. 8.1.8, p. 42] is used to make the following substitutions

$$\bar{\boldsymbol{\mu}}_{h_1h_2} = \bar{\boldsymbol{\Sigma}}_{h_1h_2}(\tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_1}\tilde{\boldsymbol{\mu}}_{\perp,h_1} + \tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_2}\tilde{\boldsymbol{\mu}}_{\perp,h_2}) \tag{5.5}$$

$$\bar{\boldsymbol{\Sigma}}_{h_1h_2} = (\tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}^{-1}_{\perp\perp,h_2})^{-1} \tag{5.6}$$

Thus, the ARS integral in $\mathbf{t}_\perp$ of eq. (5.3) can be solved

$$\begin{aligned}\mathscr{S}[f](\mathbf{U}_\perp) &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right) \\ &\quad \int_{\mathbb{R}^{d-p}} \mathrm{n}\left(\mathbf{t}_\perp - \bar{\boldsymbol{\mu}}_{h_1h_2}, \bar{\boldsymbol{\Sigma}}_{h_1h_2}\right)\, \mathbf{dt}_\perp \\ &= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{n}\left(\tilde{\boldsymbol{\mu}}_{\perp,h_1} - \tilde{\boldsymbol{\mu}}_{\perp,h_2}, \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_1} + \tilde{\boldsymbol{\Sigma}}_{\perp\perp,h_2}\right)\end{aligned} \tag{5.7}$$

where the integral of a Gaussian-like function over the whole domain is equal to 1 due to normalization.

The above derivation holds for general GMMs for generic space dimension d and Radon Transform for hyperplane of dimension p. **Focusing on the isotropic GMM case** that has been introduced before, the ARS yields

$$\mathscr{S}[f](\theta,\varphi) = \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{n}\left((\boldsymbol{\mu}_{h_2} - \boldsymbol{\mu}_{h_1})^\top \mathbf{u}_1(\theta,\varphi), \sigma^2_{h_1} + \sigma^2_{h_2}\right) \tag{5.8}$$

$$= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{e}^{-\frac{(\boldsymbol{\mu}^\top_{h_1h_2}\mathbf{u}_1)^2}{2(\sigma^2_{h_1}+\sigma^2_{h_2})}} \tag{5.9}$$

$$= \sum_{1\leqslant h_1,h_2\leqslant n_p} w_{h_1}w_{h_2}\mathrm{e}^{-\frac{\mu^2_{h_1h_2}}{4(\sigma^2_{h_1}+\sigma^2_{h_2})}(1+\cos 2\gamma_{h_1h_2})} \tag{5.10}$$

where the brief notation $\boldsymbol{\mu}_{h_1h_2} = \boldsymbol{\mu}_{h_2} - \boldsymbol{\mu}_{h_1}$, $\mu_{h_1h_2} = \|\boldsymbol{\mu}_{h_1h_2}\|$ has been adopted, and $\gamma_{h_1h_2}$ is the angle between $\boldsymbol{\mu}_{h_1h_2}$ and $\mathbf{u}_1$. Hence, there is an ARS kernel for any pair (h_1,h_2) of GMM kernels with notation

$$\mathscr{S}[f_{h_1h_2}](\theta,\varphi) = \mathrm{e}^{-\lambda_{h_1h_2}(1+\cos 2\gamma_{h_1h_2})} \tag{5.11}$$

$$\lambda_{h1h_2} = \frac{\mu^2_{h_1h_2}}{4(\sigma^2_{h_1}+\sigma^2_{h_2})} \tag{5.12}$$

The equation of ARS kernel is identical to the one given in [19], if the Gaussian mixture kernels are also identical $\sigma_h = \sigma$ for all $h = 1, \ldots, n_p$. The kernel $\mathscr{S}[f_{h_1 h_2}]$ will be expanded in spherical harmonics (SH). In the following, both a complex and a real derivation of Spherical Harmonics are described. The complex SH derivation is arguably the more elegant and commonly found in literature. However, since many coding SH libraries tend to use real SH, it is important to give a proper understanding of both.

5.1 Complex Spherical Harmonics of ARS kernels

This section derives the closed-form expression of complex spherical harmonics (CSH) expansion of ARS kernel in eq. (5.11). Please note that in the next derivation all subscript indices are dropped for simplicity.

5.1.1 Fourier series of Spherical Phase

The first step is the expansion of ARS kernel with respect to spherical phase γ. The exponential of a cosine has an elegant expression into *Fourier series* [111, (9.6.34)]. Eq. (5.11) yields

$$\begin{aligned}
\mathscr{S}[f_{h_1 h_2}](\theta, \varphi) &= \mathrm{e}^{-\lambda}\ \mathrm{e}^{-\lambda\ \cos 2\gamma} = \mathrm{e}^{-\lambda}\ \mathrm{e}^{\lambda\ \cos(2\gamma - \pi)} \\
&= \mathrm{e}^{-\lambda} \left(\mathscr{I}_0(\lambda) + 2 \sum_{k=1}^{\infty} \mathscr{I}_k(\lambda)\ \cos(2k\gamma - k\pi) \right) \\
&= \mathrm{e}^{-\lambda}\, \mathscr{I}_0(\lambda) + \sum_{k=1}^{\infty} (-1)^k 2\mathrm{e}^{-\lambda}\ \mathscr{I}_k(\lambda)\ \cos(2k\gamma) \\
&= \sum_{k=0}^{\infty} b_k\ \cos(2k\gamma) \qquad (5.13)
\end{aligned}$$

where $\mathscr{I}_k(\cdot)$ is the modified Bessel function of the first kind, γ is related to θ and φ according to eq. (5.2), and

$$b_k = \begin{cases} \mathrm{e}^{-\lambda} \mathscr{I}_0(\lambda) & k = 0 \\ (-1)^k 2\mathrm{e}^{-\lambda}\ \mathscr{I}_k(\lambda) & k > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (5.14)$$

The function $2e^{-\lambda}\ \mathscr{I}_k(\lambda)$ has been called *product of negative exponential and Bessel I* (PNEBI), its bounded and can be computed through recurrent relations.

The cosine whose argument is the multiple of a given angle can be written as the Tchebychev polynomial of the cosine

$$\cos(2k\gamma) = T_{2k}(\cos\gamma) \tag{5.15}$$

An important note is that only even orders of Fourier series have non-zero coefficients. The first consequence is that the periodicity of ARS kernel w.r.t. spherical phase γ is π and not 2π. Therefore, the observability of rotation angle on a rotated GMM in 2D ARS is up to an angle π. Similar effects are expected on observability of ARS3D. All these facts are exploited in the next section.

5.1.2 Legendre series of ARS kernel

The ARS kernel has been written as Tchebychev polynomial series. Appendix section 1.1 derives the connection coefficients for writing Tchebychev polynomials w.r.t. Legendre polynomials

$$\begin{aligned} T_{2k}(\cos\gamma) &= \sum_{l=0}^{k} \ell_{2k,l} P_{2k-2l}(\cos\gamma) \\ &= \sum_{l=0}^{k} \ell_{2k,k-l} P_{2l}(\cos\gamma) \end{aligned} \tag{5.16}$$

where $P_{2l}(x)$ is the Legendre polynomial with order $2l$. $k-l \to l$ has been changed reusing symbol l in order to have a simplified subscript index $2l$ for Legendre polynomial in the above summation. The *connection coefficients* $\ell_{2k,k-l}$ are derived in the Appendix Section 1.1 in eq. A.11.

5.1.3 Using Legendre Addition Theorem

The addition theorem of spherical harmonics (see e.g. [112]) states that

$$P_l(\cos\gamma) = \frac{4\pi}{2l+1} \sum_{m=-l}^{+l} Y_l^m(\theta,\varphi)\, \bar{Y}_l^m(\theta',\varphi') \tag{5.17}$$

where

$$\cos\gamma = \cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\varphi - \varphi') \tag{5.18}$$

In the case of ARS kernel $\mathscr{S}[f_{h_1h_2}]$, the polar parameters θ' and φ' are those of vector $\boldsymbol{\mu}_{h_1h_2} = \boldsymbol{\mu}_{h_2} - \boldsymbol{\mu}_{h_1}$, i.e.

$$\varphi' = \varphi_{h_1h_2} = \operatorname{atan2}\left(\mu_{h_1h_2,y}, \mu_{h_1h_2,x}\right) \tag{5.19}$$

$$\theta' = \theta_{h_1h_2} = \operatorname{atan2}\left(\sqrt{\mu^2_{h_1h_2,x} + \mu^2_{h_1h_2,y}}, \mu_{h_1h_2,z}\right) \tag{5.20}$$

In the following, θ' and φ' will be used, and the indices h_1 and h_2 will be omitted. Using only the even indices l, it follows that:

$$P_{2l}(\cos\gamma) = \frac{4\pi}{2(2l)+1} \sum_{m=-2l}^{+2l} Y^m_{2l}(\theta,\phi)\, \bar{Y}^m_{2l}(\theta',\phi') \tag{5.21}$$

Putting everything together, the ARS kernel is expressed as

$$\begin{aligned}
\mathscr{S}[f_{h_1h_2}](\theta,\varphi) &= \sum_{k=0}^{\infty} b_k\, T_{2k}(\cos\gamma) = \sum_{k=0}^{\infty} b_k \sum_{l=0}^{k} \ell_{2k,k-l} P_{2l}(\cos\gamma) \\
&= \sum_{k\geqslant 0, 0\leqslant l\leqslant k} b_k \ell_{2k,k-l} P_{2l}(\cos\gamma) = \sum_{l\geqslant 0, k\geqslant l} b_k \ell_{2k,k-l} P_{2l}(\cos\gamma) \\
&= \sum_{l=0}^{\infty} P_{2l}(\cos\gamma) \underbrace{\sum_{k=l}^{\infty} b_k \ell_{2k,k-l}}_{\nu_{2l}} \\
&= \sum_{l=0}^{\infty} \nu_{2l} P_{2l}(\cos\gamma)
\end{aligned} \tag{5.22}$$

where coefficients ν_{2l} are come out of a linear combination of Legendre polynomials (with even degree). The application of addition theorem and the substitution of

Legendre polynomials $P_{2l}(\cos\gamma)$ with spherical harmonics yield

$$\begin{aligned}\mathscr{S}[f_{h_1h_2}](\theta,\varphi) &= \sum_{l=0}^{\infty} \nu_{2l} \frac{4\pi}{2(2l)+1} \sum_{m=-2l}^{+2l} Y_{2l}^m(\theta,\phi)\, \bar{Y}_{2l}^m(\theta',\phi') \\ &= \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} \underbrace{\frac{4\pi\ \nu_{2l}}{4l+1} \bar{Y}_{2l}^m(\theta',\phi')}_{a_{2l,m}}\ Y_{2l}^m(\theta,\phi) \\ &= \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} a_{2l,m}\, Y_{2l}^m(\theta,\phi) \end{aligned} \tag{5.23}$$

5.1.4 Spherical Harmonic Expansion of Gaussian Mixture

Once there is the spherical harmonics expansion of each ARS kernel, the spherical harmonic expansion of a GMM is trivial, but it can still be useful to write it explicitly. First, the coefficients $a_{2l,m}$ in eq. (5.23) are renamed by adding a superscript for the point pair (h_1,h_2) as $a_{2l,m}^{h_1h_2}$. Then, the spherical harmonic series is

$$\begin{aligned}\mathscr{S}[f](\theta,\varphi) &= \sum_{h_1=1}^{n_p} \sum_{h_2=1}^{n_p} \mathscr{S}[f_{h_1h_2}](\theta,\varphi) \\ &= \sum_{h_1=1}^{n_p} \sum_{h_2=1}^{n_p} \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} a_{2l,m}^{h_1h_2}\, Y_{2l}^m(\theta,\phi) \\ &= \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} a_{2l,m}^{gmm}\, Y_{2l}^m(\theta,\phi) \end{aligned} \tag{5.24}$$

where

$$a_{2l,m}^{gmm} = \sum_{h_1=1}^{n_p} \sum_{h_2=1}^{n_p} a_{2l,m}^{h_1h_2} \tag{5.25}$$

The indices h_1 and h_2 are exchangeable, as $a_{2l,m}^{h_1h_2} = a_{2l,m}^{h_2h_1}$. In order to avoid duplicates, it is enough to iterate over $h_2 = h_1+1,\dots n_p$. Moreover, in the isotropic case, the terms with $h_1 = h_2$ are constant. These facts have been already observed also for 2D ARS [19] and are just recalled here.

5.2 Real Spherical Harmonics of ARS kernels

This section provides a similar derivation of the closed-form expression of real spherical harmonics expansion of ARS kernel.

5.2.1 Using Legendre Real Addition Theorem for Real Spherical Harmonics

The addition theorem of complex spherical harmonics [112] derives from the same theorem stated for Gegenbauer polynomials [113]. It provides the formula

$$P_l(\cos\gamma) = \frac{4\pi}{2l+1} \sum_{m=-l}^{+l} Y_l^m(\theta,\varphi)\, \bar{Y}_l^m(\theta',\varphi') \tag{5.26}$$

which connects the Legendre polynomial with the CSH.

Next, the formula can be rewritten using eq. (A.20). The three different cases according to the sign of index m are separates. For $m = 0$,

$$Y_l^0(\theta,\varphi) = Y_{l0}(\theta,\varphi) \tag{5.27}$$

For $m < 0$ (the index $|m| = -m$ and $-|m| = m$)

$$\begin{aligned}
&Y_l^m(\theta,\varphi)\bar{Y}_l^m(\theta',\varphi') = \\
&= \frac{1}{\sqrt{2}}(Y_{l,|m|}(\theta,\varphi) - iY_{l,-|m|}(\theta,\varphi))\overline{\frac{1}{\sqrt{2}}(Y_{l,|m|}(\theta',\varphi') - iY_{l,-|m|}(\theta',\varphi'))} \\
&= \frac{1}{2}((Y_{l,|m|}(\theta',\varphi') + iY_{l,-|m|}(\theta',\varphi')))(Y_{l,|m|}(\theta,\varphi) - iY_{l,-|m|}(\theta,\varphi)) \\
&= \frac{1}{2}(Y_{l,|m|}(\theta',\varphi') + iY_{l,-|m|}(\theta',\varphi'))\, Y_{l,|m|}(\theta,\varphi) \\
&+ \frac{1}{2}(Y_{l,-|m|}(\theta',\varphi') - iY_{l,|m|}(\theta,\varphi))\, Y_{l,-|m|}(\theta,\varphi)
\end{aligned} \tag{5.28}$$

For $m > 0$ (the index $|m| = m$ and $-|m| = -m$)

$$\begin{aligned}
&Y_l^m(\theta,\varphi)\,\bar{Y}_l^m(\theta',\varphi') \\
&= \frac{(-1)^m}{\sqrt{2}}(Y_{l,|m|}(\theta,\varphi) + iY_{l,-|m|}(\theta,\varphi))\overline{\frac{(-1)^m}{\sqrt{2}}(Y_{l,|m|}(\theta',\varphi') + iY_{l,-|m|}(\theta',\varphi'))} \\
&= \frac{1}{2}(Y_{l,|m|}(\theta',\varphi') - iY_{l,-|m|}(\theta',\varphi'))(Y_{l,|m|}(\theta,\varphi) + iY_{l,-|m|}(\theta,\varphi)) \\
&= \frac{1}{2}(Y_{l,|m|}(\theta',\varphi') - iY_{l,-|m|}(\theta',\varphi'))\,Y_{l,|m|}(\theta,\varphi) \\
&+ \frac{1}{2}(Y_{l,-|m|}(\theta',\varphi') + iY_{l,|m|}(\theta',\varphi'))\,Y_{l,-|m|}(\theta,\varphi)
\end{aligned} \tag{5.29}$$

Given the fixed $m > 0$, the sum of the paired terms for m and $-m$ yields

$$\begin{aligned}
&Y_l^{-m}(\theta,\varphi)\bar{Y}_l^{-m}(\theta',\varphi') + Y_l^m(\theta,\varphi)\,\bar{Y}_l^m(\theta',\varphi') = \\
&= Y_{l,-|m|}(\theta,\varphi)Y_{l,-|m|}(\theta',\varphi') + Y_{l,|m|}(\theta,\varphi)Y_{l,|m|}(\theta',\varphi') \\
&= Y_{l,-m}(\theta,\varphi)Y_{l,-m}(\theta',\varphi') + Y_{l,m}(\theta,\varphi)Y_{l,m}(\theta',\varphi')
\end{aligned} \tag{5.30}$$

Thus, the addition theorem of RSH is formulated as

$$P_l(\cos\gamma) = \frac{4\pi}{2l+1}\sum_{m=-l}^{1} Y_{lm}(\theta,\varphi)Y_{lm}(\theta',\varphi') \tag{5.31}$$

which can be rewritten once again using the addition theorem, but with real spherical harmonics as per eq. (5.31). Note that only even indices $2l$ occurring in ARS (can and) have to be taken into account

$$P_{2l}(\cos\gamma) = \frac{4\pi}{2(2l)+1}\sum_{m=-l}^{1} Y_{2l,m}(\theta,\varphi)Y_{2l,m}(\theta',\varphi') \tag{5.32}$$

5.3 Correlation of ARS3D

Also in the 3D case, the main application of ARS is the computation of rotation between two point clouds representing the same scene. Let $f_s(\mathbf{r})$ and $f_t(\mathbf{r})$ be two GMMs representing respectively the *source* and *destination* point clouds. Their ARS

are expanded into spherical harmonics as

$$\mathscr{S}[f_s](\theta,\varphi) = \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} a^s_{2l,m}\, Y^m_{2l}(\theta,\phi) \tag{5.33}$$

$$\mathscr{S}[f_d](\theta,\varphi) = \sum_{l=0}^{\infty} \sum_{m=-2l}^{+2l} a^d_{2l,m}\, Y^m_{2l}(\theta,\phi) \tag{5.34}$$

where superscripts ^{s} and ^{d} or subscripts $_s$ and $_d$ label source and destination terms.

An important property of ARS that has been shown in Section 3.2 is its translation invariance. Assume that $f_s(\mathbf{r}) = f_t(\mathbf{R}\mathbf{r}+\mathbf{t})$ for some rotation $\mathbf{R} \in SO(d)$ and translation $\mathbf{t} \in \mathbb{R}^d$ (in our case $d = 3$). Hence, $f_s(\mathbf{r})$ is a transformed copy of $f_t(\mathbf{r})$. Due to translation invariance, the ARS does not change when translated. Now, concentrating on the estimation of rotation, it is important to define a rotation operator $\mathscr{R}$ s.t.

$$\mathscr{R}\mathscr{S}[f(\mathbf{r})](\theta,\varphi) = \mathscr{S}\left[f(\mathbf{R}^{-1}\mathbf{r})\right](\theta,\varphi) \tag{5.35}$$

The notation distinguishes between the operator $\mathscr{R}$ and the matrix $\mathbf{R}$, but they are of course closely associated. The ARS $\mathscr{R}\mathscr{S}[f]$ of the rotated point cloud can be written w.r.t. the spherical harmonics and the coefficients using RSH matrices. The expression of the rotated $Y^m_l(\theta,\varphi)$ is

$$\mathscr{R}Y^m_l(\theta,\varphi) = \sum_{m'=-l}^{l} (D^l_{mm'})^*\, Y^{m'}_l(\theta,\varphi) \tag{5.36}$$

where $D^l_{mm'}$ is the Wigner D-matrix [114, 115, 116]. Actually, the name matrix refers to the relation between the $2l+1$ rotated harmonics $\mathscr{R}Y^m_l(\theta,\varphi)$ with fixed index l and $m = -l,\ldots,l$ and their (non-rotated) counterparts $Y^{m'}_l(\theta,\varphi)$. For a given l, they are related by the $(2l+1)\times(2l+1)$ matrix $D^l_{mm'}$ where m is the row index and m' the column index.

$D^l_{mm'}$ is a function of the rotation that can be represented in several ways. The most common is ZYZ Euler angles (α,β,γ), but other parameters like quaternions have been adopted too. Pleaser refer to the appendix (chapter A) for a more in-depth discussion. In this section it can simply be assumed that $D^l_{mm'}$ depends on rotation

parameters. Also, the parameters can be simply referred to by using the rotation matrix **R** until a specific choice is required. The rotated ARS of source point cloud can be written using eq. (5.36) as

$$\mathscr{R}\mathscr{S}\left[f(\mathbf{r})\right](\theta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-2l}^{+2l} a^{s}_{2l,m}\,\mathscr{R}Y^{m}_{2l}(\theta,\phi) \tag{5.37}$$

$$= \sum_{l=0}^{\infty}\sum_{m=-2l}^{+2l} a^{s}_{2l,m}\sum_{m'=-2l}^{2l}(D^{l}_{mm'})^{*}\,Y_{l}^{m'}(\theta,\varphi) \tag{5.38}$$

$$\tag{5.39}$$

The *convolution of ARS* is defined as follows:

$$\begin{aligned} C(\mathbf{R}) &= \mathscr{S}\left[f_d\right] \circledast \mathscr{S}\left[f_s\right] \\ &= \int_0^{\pi}\int_0^{2\pi} \mathscr{S}\left[f_d\right](\theta,\varphi)\,\mathscr{R}\mathscr{S}\left[f_s\right]^{*}(\theta,\varphi)\sin\varphi d\varphi d\theta \end{aligned} \tag{5.40}$$

where $\mathscr{R}$ is the rotation operator associated to rotation **R** as before. The meaning of this operation is clear. If the source and destination point clouds represents the same scene from different viewpoints, then $\mathscr{S}\left[f_s\right](\theta,\varphi)$ is a rotated copy of $\mathscr{S}\left[f_d\right](\theta,\varphi)$ up to non-corresponding points or measurement noise. Thus, the best estimation of rotation $\mathbf{R}^{*}$ between the two viewpoints corresponds to the maximum of correlation, when the rotated source ARS $\mathscr{R}\mathscr{S}\left[f_s\right](\theta,\varphi)$ overlaps with the destination ARS.

The correlation is expanded using the equations (5.33) and (5.34), the rotated RSH matrices as reported in eq. (5.36) and the orthogonality of spherical harmonics defined as

$$\int_0^{\pi}\int_0^{2\pi}(Y_l^m(\theta,\varphi))^{*}\,Y_{l'}^{m'}(\theta,\varphi)\,\sin\varphi d\varphi d\theta = \frac{4\pi}{2l+1}\,\delta_{ll'}\,\delta_{mm'} \tag{5.41}$$

where $\delta_{ll'}$ and $\delta_{mm'}$ are Kronecker deltas. The term $\sqrt{4\pi/(2l+1)}$ is sometimes included as part of Racah normalization coefficient in the definition of Y_l^m, but not in

this document. Thus, the correlation yields

$$\begin{aligned} C(\mathbf{R}) &= \sum_{l=0}^{\infty} \sum_{m=-2l}^{2l} \sum_{l'=0}^{\infty} \sum_{m'=-2l'}^{2l'} \sum_{m"=-2l'}^{2l'} a^d_{2l,m} \, (a^s_{2l',m'})^* \, D^{2l'}_{m'm"} \\ &\cdot \int_0^{\pi} \int_0^{2\pi} Y^m_{2l}(\theta,\varphi) \, Y^{m"}_{2l'}(\theta,\varphi) \sin\varphi d\varphi d\theta \\ &= \sum_{l=0}^{\infty} \sum_{m'=-2l}^{2l} \sum_{m=-2l}^{2l} \frac{4\pi \, a^d_{2l,m} \, (a^s_{2l,m'})^*}{2l+1} \, D^{2l}_{mm'}(\mathbf{R}) \\ &= \sum_{l=0}^{\infty} \sum_{m'=-2l}^{2l} \sum_{m=-2l}^{2l} a^c_{2l,m,m'} D^{2l}_{mm'}(\mathbf{R}) \end{aligned} \tag{5.42}$$

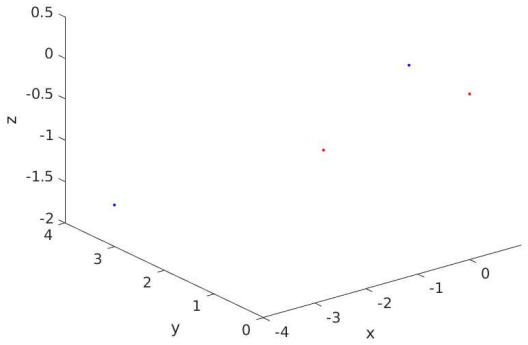
where all terms with $l' \neq l$ and $m" \neq m$ have been removed, and the dependence of RSH matrices from rotation $\mathbf{R}$ have been explicitly added. Also the *convolution coefficients* used above have the following expressions:

$$a^c_{2l,m,m'} = \frac{4\pi \, a^d_{2l,m} \, (a^s_{2l,m'})^*}{2l+1} \tag{5.43}$$

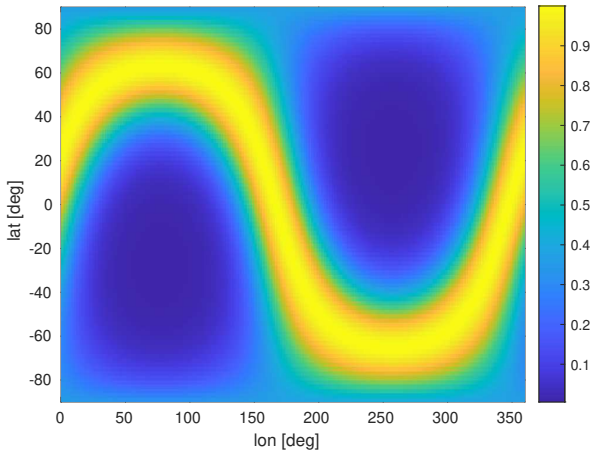
In Figures 5.1 and 5.2, it can be seen how the roto-translation impacts the spectrum of a point cloud. As a matter of fact, a shift in the maximum of the source ARS spectrum of 60° can be observed, which corresponds to the rotation linking source and destination point clouds.

5.4 Rotation Estimation

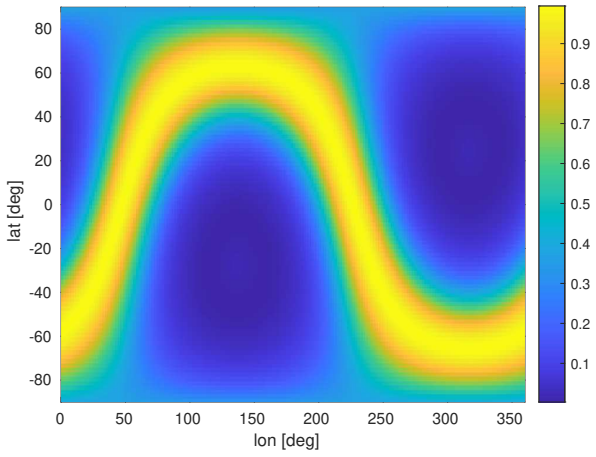
In a similar fashion to the planar case, the correlation function presented in the previous section can be used to align a *source* and a *destination* point cloud. In particular, the correlation function defined in eq. (5.42) can be used as a cost function to be maximized. Rotation matrix $\mathbf{R}^*$, for which the correlation function assumes the maximum value, is the output of the ARS3D rotation estimation pipeline. So far, two methods have been proposed for performing the maximum optimization of the correlation function. The first method is based on a simple grid sampling through ZYZ-Euler angles (proposed in [117] and inspired by [118]) angles of all possible



(a) Src (red) and Dst (blue) Two Point Clouds

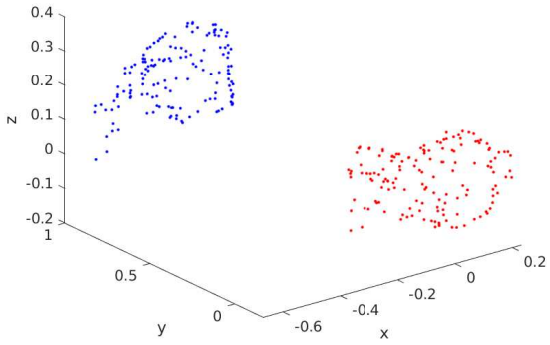


(b) Source Cloud Spectrum

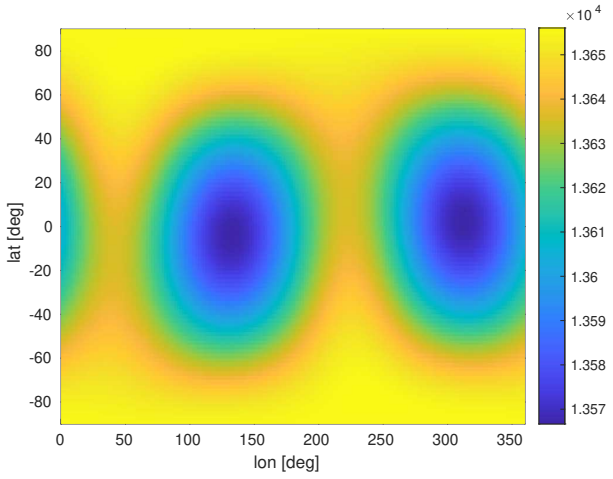


(c) Destination Cloud Spectrum

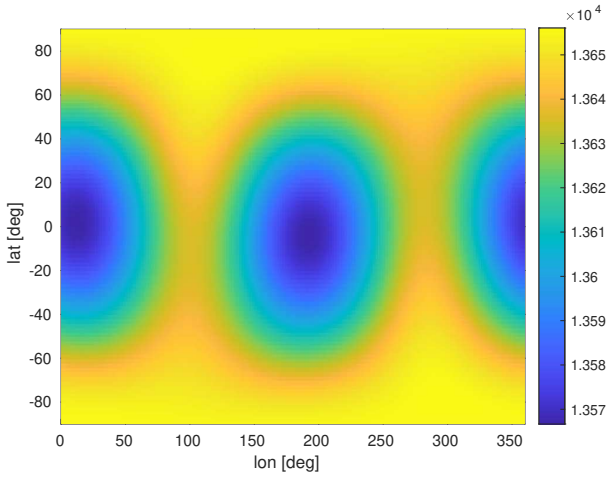
Figure 5.1: ARS3D spectrum effect (60° rotation, 2 point cloud)
lon, *lat* are respectively the φ, θ parameters of the spectra



(a) Src (red) and Dst (blue) Bunny Clouds



(b) Source Cloud Spectrum



(c) Destination Cloud Spectrum

Figure 5.2: ARS3D spectrum effect (60° rotation, Bunny cloud)
lon, *lat* are respectively the φ, θ parameters of the spectra

rotations. Briefly, this grid is composed through regular sampling in terms of ZYZ Euler angles (α, β, γ) such that

$$\alpha_j = \gamma_j = \frac{2\pi j}{2B} \tag{5.44}$$

$$\beta_k = \frac{\pi(2k+1)}{4B} \tag{5.45}$$

$$\text{with } 0 \leq j,k < 2B$$

According to [119], this sampling scheme is suitable for the analysis of bandlimited functions with bandwidth B. With this sampling, the size of the search space is $2B \times 2B \times 2B$. The grid sampling is then paired with a brute-force search on the grid. Empirically, the desired accuracy has been obtained by sampling with a 1° resolution on each of the three angles, but more implementation details will be discussed in Section 5.5.

The second method is through the use of Riemannian geometry optimization. To perform this optimization, an element-wise expression for the Euclidean gradient has been computed, as explained in Section 1.3.1 of the appendix. The (complex) Wigner matrices mentioned previously are the most common notation for spherical harmonics. However, for the specific problem tackled here, due to the Hermitian symmetry of the complex coefficients of a real function, the more practical Real Spherical Harmonics (RSH) have been used. A common notation for RSH is $\mathbf{M}_l^z(\gamma)$, where M is an array of matrices, subscript l indicates the order of the matrix composing the SH, superscript z is the rotation axis and γ the rotation angle. Another notation used is $\mathbf{M}_l(E)$, where E is a rotation matrix and the associated RSH is obtained through axis-wise composition. For more details on the relation between rotations and SH, please refer to Section 1.2 in the appendix. Reporting only the results here, the expressions

for the RSH matrices in the case regarding rotations around the Z-axis are:

$$\mathbf{M}_l^z(\gamma) = \mathbf{M}_l(\mathbf{R}_z(\gamma)) = \begin{bmatrix} \cos(l\gamma) & \cdots & 0 & 0 & 0 & 0 & \sin(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & \cos\gamma & 0 & \sin\gamma & & 0 \\ 0 & & 0 & 1 & 0 & & 0 \\ 0 & & -\sin\gamma & 0 & \cos\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ -\sin(l\gamma) & 0 & 0 & 0 & 0 & \cdots & \cos(l\gamma) \end{bmatrix} \tag{5.46}$$

with their element-wise first and second derivatives being:

$$\dot{\mathbf{M}}_l^z(\gamma) = \begin{bmatrix} -l\sin(l\gamma) & \cdots & 0 & 0 & 0 & 0 & l\cos(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & -\sin\gamma & 0 & \cos\gamma & & 0 \\ 0 & & 0 & 0 & 0 & & 0 \\ 0 & & -\cos\gamma & 0 & -\sin\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ -l\cos(l\gamma) & 0 & 0 & 0 & 0 & \cdots & -l\sin(l\gamma) \end{bmatrix} \tag{5.47}$$

$$\ddot{\mathbf{M}}_l^z(\gamma) = \begin{bmatrix} -l^2\cos(l\gamma) & \cdots & 0 & 0 & 0 & 0 & -l^2\sin(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & -\cos\gamma & 0 & -\sin\gamma & & 0 \\ 0 & & 0 & 0 & 0 & & 0 \\ 0 & & \cos\gamma & 0 & -\cos\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ l^2\sin(l\gamma) & 0 & 0 & 0 & 0 & \cdots & -l^2\cos(l\gamma) \end{bmatrix}. \tag{5.48}$$

Then, rotation coefficients matrices for Y-axis rotation are handled as a permutation

Figure 5.3: *Happy*, *Lucy* and *Bunny* sample scans of the Stanford dataset.

of the Z-axis coefficients such that

$$\mathbf{M}_l(\mathbf{R}_Y(\beta)) = \mathbf{M}_l(\mathbf{E}^\top)\,\mathbf{M}_l(\mathbf{R}_Z(\beta))\,\mathbf{M}_l(\mathbf{E}) \tag{5.49}$$

$$\text{with } \mathbf{E} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.50}$$

where M_l are the rotation coefficient matrices of order l. Finally, optimization is performed through Riemannian Trust Regions [120], with the Riemannian gradient and Hessian being numerically computed by the optimization library from the Euclidean gradient provided as explained in the appendix (section 1.3.3).

5.5 ARS3D Experiments

The evaluation of the spatial version of ARS has been performed on the Stanford 3D Scanning Repository [121], composed of range data and detailed reconstructions. Examples of the scans contained in the dataset are reported in Figure 5.3. These have been sampled in order to obtain couples of point clouds on which the alignment and pose estimation procedure based on ARS has been tested. As in the 2D case, a

mandatory feature that the Stanford dataset presents is the ground truth information, which allows evaluation through pairwise confrontation of the ARS pipeline. Similarly to the 2D case, the evaluation over the Stanford dataset has focused on four major metrics:

- rotation error
- percentage of success
- standard deviation
- execution time

ARS has been compared against GO-ICP [16], Teaser [23] and Point Cloud Library's [122] registration module. Both Teaser and Point Cloud Library (PCL) have been initialized with FPFH [3] features. The error evaluation is done using the angle-axis notation on the rotation that separates the ground truth rotation from the rotation estimated with the various methods. First, the estimations are separated into positive and negative: if the angle's magnitude is less than 5°, then the rotation estimation is considered as successful (i.e., positive); otherwise the estimation is considered failed (negative). The mean error and standard deviation are then computed on the successful estimations. In a similar fashion to what has been done for the 2D tests with the MPEG7 dataset, three test sets have been constructed by introducing noise, occluding some areas of the clouds, and adding random points to simulate sensory measurement errors. The results are reported in figures 5.4, 5.5,5.6.

5.6 Discussion

This chapter has presented the 3D spatial formulation of ARS. The correlation optimization procedure through Spherical Harmonics expansion has been discussed, together with its usability in the context of 3D rotation estimation. The experimental results show clear accuracy improvements over local methods, while holding up against other global methods, but with less intensive computational time requirements. When able to complete in reasonable time, GO-ICP seems to outperform ARS3D. However,

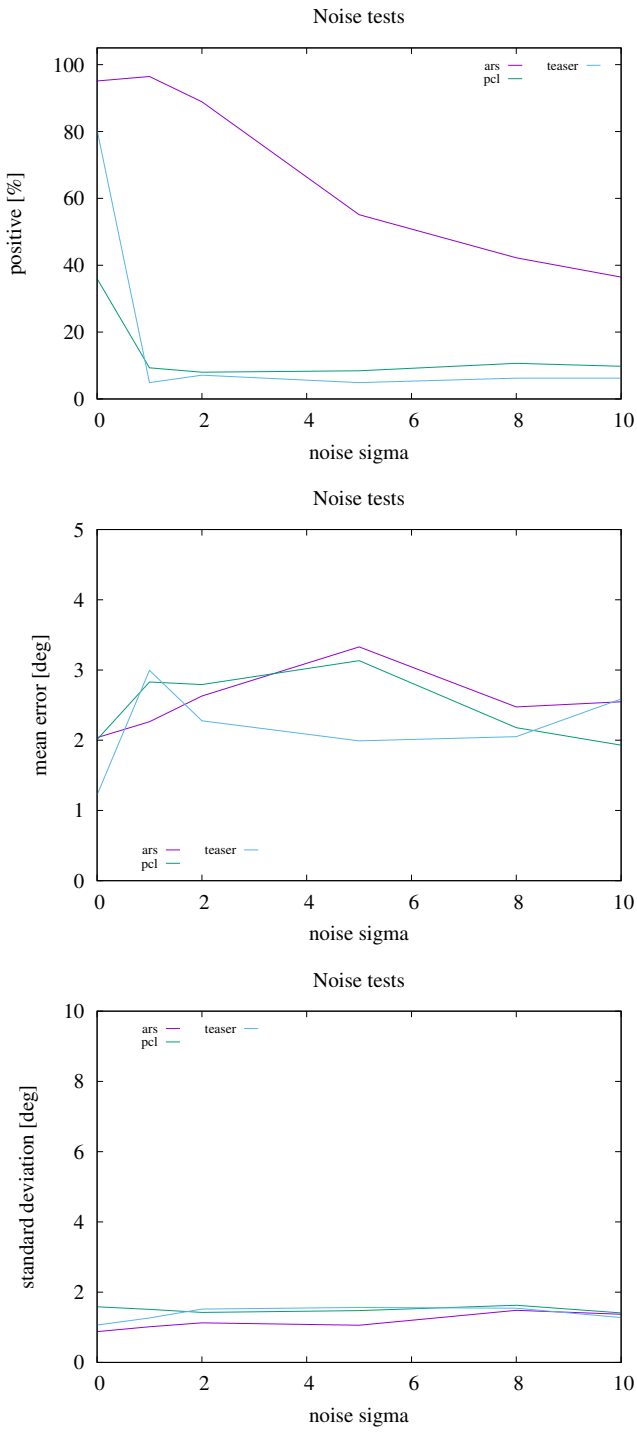


Figure 5.4: ARS3D rotation estimation results on Stanford 3D Scanning Repository dataset - noise tests

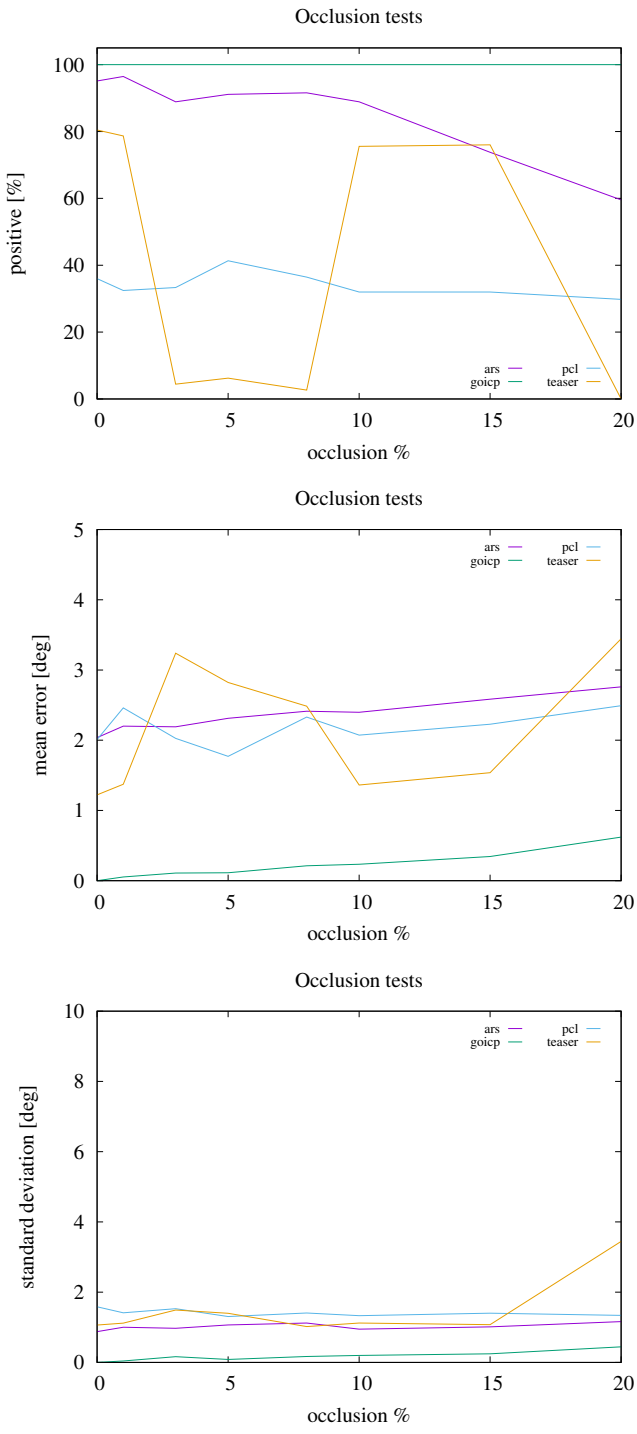


Figure 5.5: ARS3D rotation estimation results on Stanford 3D Scanning Repository dataset - occlusion tests

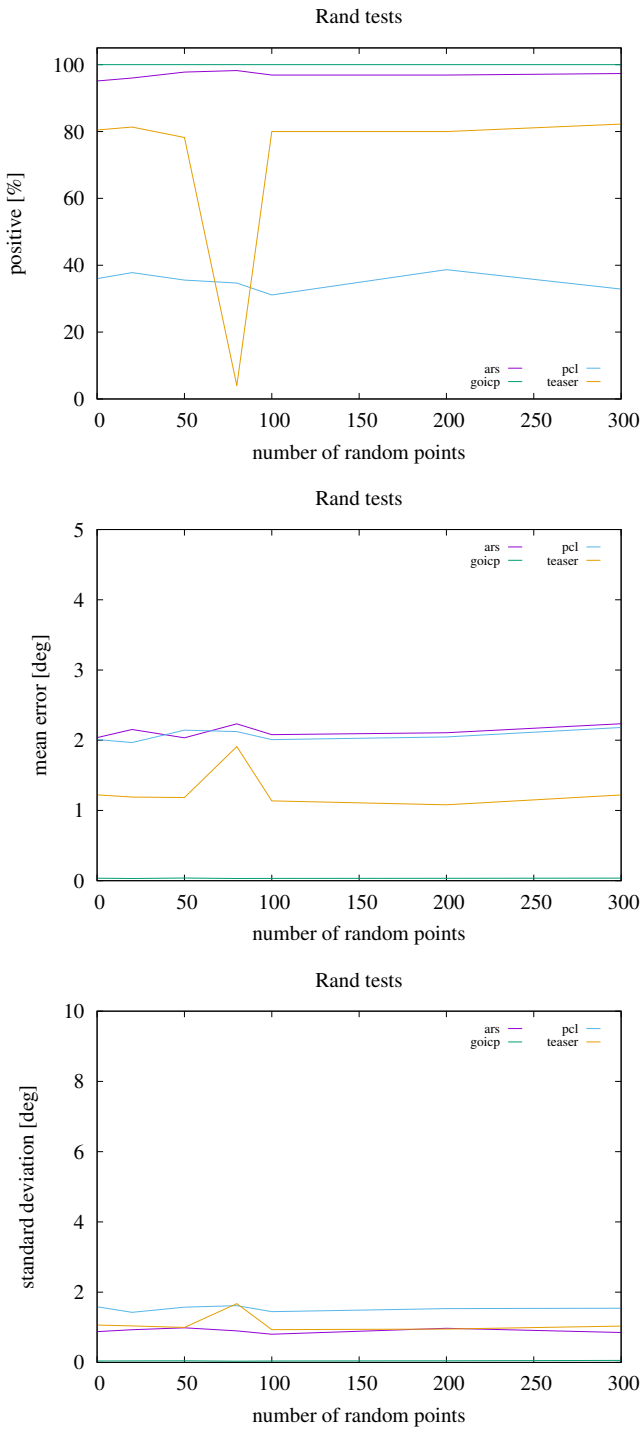


Figure 5.6: ARS3D rotation estimation results on Stanford 3D Scanning Repository dataset - random point tests

it can be noted the absence of GO-ICP from the noise tests (see Figure 5.4) as the method could not complete even a single evaluation with noise $\sigma > 0$, due to the huge time complexity increase.

In general, it is safe to say that the presence of a more robust cost optimization procedure (for rotation estimation) could bring benefits in both the accuracy of results in the harder test cases, as well as a noticeable reduction of the execution times. Additional tests on more realistic datasets (i.e., extracted from 3D sensors) will help with the assessment of the quality of ARS3D performance and adaptability.

Chapter 6

Pose Averaging through Shape of Motion (SOM) Matrix Formulation

This chapter addresses the problem of 3D pose estimation w.r.t. a world reference frame, given a set of relative pairwise observations. This problem arises within multi-agent systems that perform cross-localization, or more in general when a set of sensors need to estimate the relative pose of all members that compose the fleet. In this context, cross-localization is the name chosen to specify the particular case of the pose averaging problem (recall Chapter 2) of interest here. The problem of cross-localization inside a distributed system of cameras is a relevant problem that sees one of its most notable use cases in a fleet of drones. The relation of this problem with the pose graph formulation to solve SLAM or the bundle adjustment problem has been emphasized during the discussion of Chapter 2.

Now, the *Shape of Motion* (acronym SOM) matrix [123, 124] is a commonly used tool for representing sets of N rotations and translations, for example in a distributed system for localization. Given $\mathbf{R} = [R_1, R_2, \ldots, R_N]$ and $\mathbf{T} = [T_1, T_2, \ldots, T_N, I]$,

the most common way to represent the Shape of Motion matrix W is as follows:

$$W = \begin{bmatrix} R_1^\top T_1 & R_1^\top T_2 & \cdots & R_1^\top T_N & R_1^\top \\ R_2^\top T_1 & R_2^\top T_2 & \cdots & R_2^\top T_N & R_2^\top \\ \vdots & \vdots & \ddots & \vdots & \\ R_N^\top T_1 & R_N^\top T_2 & \cdots & R_N^\top T_N & R_N^\top \end{bmatrix} = \mathbf{R}^\top \mathbf{T} \,. \tag{6.1}$$

The following formulation based on the Shape of Motion matrix can be used to solve the cross-localization problem (with acronym CL-SOM, schematized in Figure 6.1). As a matter of fact, the blocks composing block-matrix W are used to formulate the cost function to be minimized, at it is apparent from upcoming eq. (6.4). Let $G = (V,E)$ be a graph where nodes in V represent sensors or cameras, each with a given pose $\{R_i, T_i\}$. More details on the reference frame in which these node poses are expressed can be found in Section 6.1. The nodes are able to obtain/compute and exchange information about their pairwise estimated translation. The edges represent such ${}^iT_{ij} \in \mathbb{R}^d$ information i.e., the translation that links cameras i and j with respect to camera i's frame. The graph does not necessarily have to be complete. Also, the information about rotations and translations of *each* camera with reference to a given frame (again, see Section 6.1 for a more detailed explanation of which frame is this) is given in the form of the following block matrices

$$\mathbf{R} = [R_1, R_2, ..., R_N] \in SO(d)^N \tag{6.2}$$

$$\mathbf{T} = [T_1, T_2, ..., T_N] \in \mathbb{R}^{d \times n} \,. \tag{6.3}$$

Hence, the cross-localization problem can be formulated as follows. Let V be a set of N cameras with their associated poses. Let R_i be the rotation from the local reference frame to the world reference frame, and let T_i be the translation in the world reference frame, $i \in V$. Given the set E of the relative translation measurements between couples of cameras (i, j), the problem becomes an optimization problem with the following cost function:

$$\min_{R,T} \sum_{i,j \in E} ||{}^iT_{ij} - R_i^\top T_j + R_i^\top T_i||^2 \tag{6.4}$$

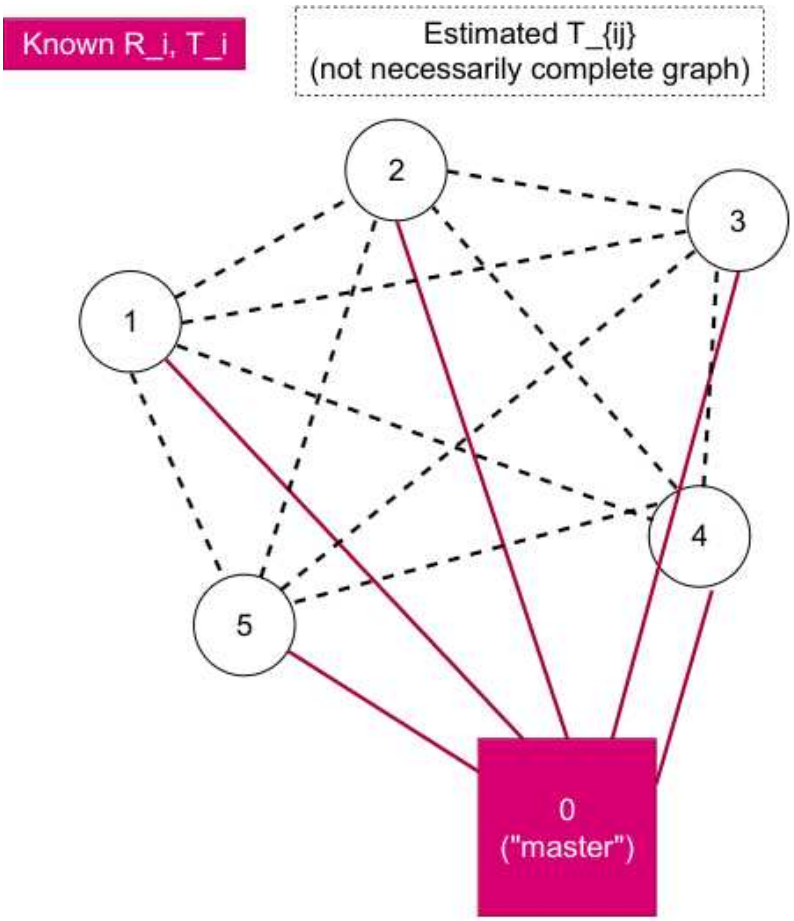


Figure 6.1: Cross-Localization of Sensors: Problem Setup Graph

to be minimized. As previously mentioned, the terms $R_i^\top, T_i$ in eq. (6.4) are the block elements of the Shape of Motion matrix W in eq. (6.1). Now, the presented problem will be formulated in order to be solved as a two-step procedure, with each step contributing to solving the rotation part of the pose and, respectively, the translation. It has to be noted that the problem separates over R_i,

$$\sum_i \min_{R_i} \sum_j ||{}^iT_{ij} - R_i^\top (T_j - T_i)||^2 \tag{6.5}$$

and becomes an orthogonal Procrustes problem for each R_i. The two parts of the problem can be rewritten as:

1.
$$\min_{R \in SO(d)^N} \; tr(R^\top L(T) R) + tr(R^\top P(T)) \tag{6.6}$$

2.
$$\min_{T \in \mathbb{R}^{d \times N}} \; ||A(R)vec(T) + b(R)||^2 \tag{6.7}$$

The matrices $L(T), P(T)$ in eq. (6.6) and, respectively, matrices $A(R), b(R)$ in eq. (6.7) are explicitly written as function or rotations matrix R and translations matrix T, but in the following will be simply referred to as L, P, A, b. Equations (6.6) and (6.7) are simple rearrangements of the original cost function (6.4), and their matricial terms can be trivially computed from it e.g., $L(T)_{ij} = (T_j T_j^\top - T_i T_j^\top - T_j T_i^\top + T_i T_i^\top)$ comes out of rewriting the first part of (6.6)'s cost function as $tr\left(\sum_{e_{ij}} R_i^\top L_{ij} R_i\right) = tr\left(R^\top L R\right)$. According to the common notation of block matrices, $L(T)_{ij}$ is the (i,j)-th block (with size $d \times d$) of block matrix L (with size $dN \times dN$). Also, $vec(\cdot)$ is the vectorization operator e.g., its correspondent in MATLAB code would be $(:)$.

6.1 Gauge symmetries

The constraint in (6.4) can, at best, fix the rotations and translations of each camera up to a global rotation and translation. This is due to the fact that

$$
\begin{aligned}
&{}^{i}T_{ij} - R_i^\top T_j + R_i^\top T_i = && (6.8)\\
&{}^{i}T_{ij} - R_i^\top R_0^\top (R_0(T_j + s)) + R_i^\top R_0^\top (R_0(T_i + s)) = && (6.9)\\
&\left(={}^{i}T_{ij} - R_i^\top R_0^\top (R_0 T_j + s) + R_i^\top R_0^\top (R_0 T_i + s)\right) && (6.10)
\end{aligned}
$$

for any $R_0 \in SO(3)$, $s \in \mathbb{R}^3$. This corresponds to a global rigid body transformation (R_0, s) of all the cameras where $R_i \to R_0 R_i$, $T_i \to R_0(T_i + s)$.

In general, the algorithms based on these formulations will return results with arbitrary R_0, s. To evaluate the results, three solutions can be adopted:

1. Fix the rotation and translation of one of the cameras, e.g., camera 1 s.t. $R_1 = I$, $T_1 = 0$. This however has the problem that the distance between solutions depends on which camera is chosen.

2. Evaluate only relative quantities, i.e., let R_i, T_i be the solution returned by an algorithm, and R_i^g, T_i^g a ground-truth solution. Then, evaluate

$$
\begin{aligned}
&d(R_i^\top R_j) - d({R_i^g}^\top R_j^g) && (6.11)\\
&||R_i^\top (T_i - T_j) - {R_i^g}^\top (T_i^g - T_j^g)|| && (6.12)
\end{aligned}
$$

3. Fix the gauge ambiguity by adding additional terms to the cost in eq. (6.4):

$$
\sum_{i,j \in E} ||{}^{i}T_{ij} - R_i^\top T_j + R_i^\top T_i||^2 + ||\sum_i T_i||^2 + ||\sum_i R_i - I||^2 \quad . \tag{6.13}
$$

In the following, consider that the solution adopted is always the first one i.e., one the cameras is chosen as *master*, and all R_i, T_i couples are expressed w.r.t. chosen master (see Figure 6.1) camera's frame. It has to be noted that this assumption does not cause a loss in generality of the proposed solution, as the choice of the master node is arbitrary and can be switched in consecutive executions of the algorithm.

6.2 Riemannian Gradients and Hessians

This section defines the main terms occurring in the optimization problem of eq. (6.4). Since the multiple rotations R to be estimated have manifold structure, also the gradients and Hessian matrices require proper definition on manifolds. In general, notation-wise, the overhead $\overline{bar}$ indicates a Euclidean operator, while the name without the overline indicates the Riemannian operator i.e., $\overline{grad}$ and $grad$.

6.2.1 Stiefel Manifold

Despite sets of rotations being usually represented as matrices in $SO(d)^N$, the formulation for this problem allows them to be inside the Stiefel manifold. A matrix $M \in \mathbb{R}^{p,d}$ is considered as part of the Stiefel manifold if it satisfies the

$$M^\top M = I_d \tag{6.14}$$

orthogonality constraint. The most common notation in literature for this kind of manifold is $St(d,p)$, $d \leqslant p$, with the number of columns put first in order to signal the maximum possible rank of these matrices, as well as the dimension of the identity matrix (see constraint eq. (6.14)). In short, the Stiefel Manifold contains the tall-skinny orthogonal matrices. According to a geometric interpretation, an element of the Stiefel manifold is a $p \times d$ matrix whose columns are d orthogonal vectors in space $\mathbb{R}^p$. Their importance is clear in problems involving multiple rotations where orthogonality constraints hold for a subset of vectors with cardinality less than the dimension of the space.

Another important notation is the one for 3D Stiefel manifold i.e., $St(nrs,d,N)$ or $St(d,nrs)^N$. In particular, a tensor $A \in St(nrs,d,N)$ will contain N matrices $\in St(d,nrs)$ spanning along the third dimension. *Note: nrs* is an acronym for number of Stiefel rows. The 3D Stiefel manifold is normally consider through its tensor representation, but in some cases also its *stacked* representation will be used. To switch representations, the $matStack()$ and $matUnstack()$ operators, which are described in the *Tensor Operators* paragraph inside Section 6.2.3).

In the upcoming subsections, gradients and Hessians used for computing retractions and other Riemannian geometry elements are computed. A good part of the theory behind the derivation of these tools can be found in [120, 125, 126, 127, 128, 129, 130, 131].

6.2.2 Gradients

A paramount concept of Riemannian geometry is the one of Riemannian gradients. Thus, first choose a Riemannian metric, and then a notion of gradient ensues. Notably, a manifold can be interpreted as multiple objects at the same time. The simplest interpretation to be given is linked to its immersion in the Euclidean space, with the classical Euclidean metric associated to each element (i.e., the Euclidean norm of the vectorized element). Still, the manifold structure can be exploited in a much more appropriate way, through the usage of Riemannian metrics. To better introduce this concept, take for example one of the simplest and most used manifolds, d-dimensional unit sphere S^{d-1}. Also, let T_xS^{d-1} be the tangent space to a given point x on the sphere. Since T_xS^{d-1} is a different linear subspace for various $x \in S^{d-1}$, a different inner product for each point is needed: $< \cdot, \cdot >_x$ denotes the particular choice of inner product on T_xS^{d-1}. If this choice of inner products varies *smoothly* [125] with x, then it is a *Riemannian metric*. S^{d-1} equipped with this metric is called a Riemannian manifold. This allows the definition of the *Riemannian gradient* of f on S^{d-1}: $grad\ f(x)$ is the unique tangent vector at x such that, $\forall v \in T_xS^{d-1}$,

$$Df(x)[v] =< v, grad\ f(x) >_x, \tag{6.15}$$

where D is the differential of f at x along v. Thus, then a notion of (Riemannian) gradient ensues from the choice of a Riemannian metric.

Euclidean gradient

Now, shift the focus towards finding a formula for the Euclidean gradient of the cost function f written in the more straightforward notation of eq. (6.6) i.e.,

$$f : St(d,m)^n \subset \mathbb{R}^{m\times d\times n} \to \mathbb{R} \tag{6.16}$$

$$x \mapsto tr(x^\top Lx + x^\top P) \ . \tag{6.17}$$

Then:

$$\overline{grad} f(x) = matUnstack((L+L^\top)x_{stacked} + P) \tag{6.18}$$

$$\tag{6.19}$$

where the $matUnstack()$ operator is used to express the gradient in the chosen 3D search space i.e., $\mathbb{R}^{m\times d\times n}$. Its inverse pair is the $matStack()$ operator that simply squeezes a 3D array $\in \mathbb{R}^{m\times d\times n}$ into a corresponding 2D dimensionality $\mathbb{R}^{mn\times d}$, and is sometimes directly expressed as:

$$x_{stacked} = matStack(x) \ . \tag{6.20}$$

Riemannian Gradient

Let now f be the cost function of eq. (6.17). Its Riemannian gradient is as follows:

$$grad\ f(x) = stiefel\_tangentProj(x, \overline{grad}(x, problem)) \tag{6.21}$$

where

$$stiefel\_tangentProj(Y,H) = H - Y * sym(Y^\top * H) \tag{6.22}$$

and $stiefel\_tangentProj()$ is applied member-wise to every matrix in the third dimension, and projects H onto the tangent space of the Stiefel manifold at Y. Also, $sym()$ is the operator that extracts the symmetric part from a square matrix i.e.,

$$A_{sym} = sym(A) = \frac{A+A^\top}{2} \tag{6.23}$$

with $A \in \mathbb{R}^{n\times n}$.

6.2.3 Hessians

In a similar fashion as what has been done for gradients in Section 6.2.2, the aim now becomes finding the expressions for the Euclidean and Riemannian Hessians of cost function reported in eq. 6.17.

Tensor Operators

A series of 3D tensor operators in used in this project. Let the $_{3D}$ subscript signal a 3D tensor.

- The already mentioned $matStack(R_{3D})$ and $matUnstack(R)$ operators.
  The $matUnstack()$ operator is used to express the gradient in the 3D search space i.e., $\mathbb{R}^{m\times d\times n}$. Its inverse pair is the $matStack()$ operator that simply squeezes a 3D array $\in \mathbb{R}^{m\times d\times n}$ into a corresponding 2D dimensionality $\mathbb{R}^{mn\times d}$ by stacking the matrices spanning along the third dimension on top of each other.

- $multitrace(A_{3D})$
  A scalar corresponding to the sum of all main diagonal elements of all matrices spanning along the third dimension of A_{3D}.

- $multiprod(A_{3D},B_{3D})$
  Let $C_{3D} = multiprod(A_{3D},B_{3D})$. The result C_{3D} is a tensor containing the matrix products of A_{3D},B_{3D} e.g., calling $A[:,:,i]$ the i-th element along the third dimension of A_{3D}, then
  $C[:,:,i] = A[:,:,i]B[:,:,i]$.

- $multitransp(A_{3D})$
  Let $B_{3D} = multitransp(A_{3D})$. Then, $B[:,:,i] = A[:,:,i]^\top$.

In the most relevant cases for the problem tackled in this dissertation, the 3D tensor has Stiefel matrices spanning along the third dimension that, when vertically stacked, form the Stiefel equivalent of block matrix $\mathbf{R}$ of SOM matrix $\mathbf{W}$ (recall eqs. (6.1),(6.2)).

Euclidean Hessian

The Euclidean Hessian can be directly derived from the application of the differential operator to the Euclidean gradient, *multiprod*-ed to u:

$$D\,\overline{grad}(f(x))[u] = D\big((L+L^\top)x+P\big)[u] = (L+L^\top)u \ . \tag{6.24}$$

Note that the u multiplication at the end is due to representation requirements. In this case, u is a vector $\in St(nrs,d,N)$ rather than as a matrix).

Riemannian Hessian

The following tools are useful for obtaining the Riemannian Hessian from the gradient: the **orthogonal projector** [125] $\mathscr{P}_x$ from $\mathscr{E}$ onto $T_xSt(p,n)$ is:

$$\begin{aligned}\mathscr{P}_x U = (\mathbb{I}-XX^\top)U + X\frac{1}{2}(X^\top U - U^\top X) = \\ U - X\frac{1}{2}(X^\top U + U^\top X)\end{aligned} \tag{6.25}$$

where, as usual, $St(p,n)\subset\mathbb{R}^{n\times p}$ and is a Riemannian submanifold of the Euclidean space $\mathbb{R}^{n\times p}$ itself. Another important notation is the one for the differentials of the projector:

$$\mathscr{P}_u \triangleq D(x\mapsto Proj_x)(x)[u] = \frac{\mathrm{d}}{\mathrm{d}t}Proj_{c(t)}\bigg|_{t=0} \tag{6.26}$$

$Proj_x$ is a linear map from the Euclidean tangent space $T_x\mathbb{R}^d$ to the manifold tangent space $T_x\mathscr{M}$. It can be thought as a matrix mapping tangent vectors to tangent vectors. $\mathscr{P}_u$ captures how this operator (matrix) changes when the base point x changes; $\mathscr{P}_u$ can then be thought as a matrix still mapping tangent vectors to tangent vectors but that depends on $\dot{x}$. So, differentiating eq. (6.25) with respect to X gives us the following expression for the differential (6.26):

$$D_x(\mathscr{P}_x U)[\dot{X}] = \dot{X}\frac{1}{2}(X^\top U + U^\top X) + X\frac{1}{2}(\dot{X}^\top U + U^\top \dot{X}) \tag{6.27}$$

Another important operator for obtaining the Riemannian Hessian is the Riemannian connection ∇. Let $\nabla_U(grad\ f)$ be the derivative of the gradient vector field of f along U. By definition, this is also the Euclidean Hessian of f along U:

$$\nabla_U(grad\ f) = Hess\ f[U] \tag{6.28}$$

with the understanding that

$$Hess\ f[U](x) = \tag{6.29}$$

$$Hess\ f(x)[U(x)] = \tag{6.30}$$

$$\nabla_{U(x)}(grad\ f) \tag{6.31}$$

The Hessian satisfies the self-adjoint property:

$$< Hess\ f[U], V >=< Hess\ f[U], V > \tag{6.32}$$

Finally, according to eqs. (6.25), (6.28), the Riemannian Hessian is as follows:

$$Hess\ f(x)[U] = x\ symm(x^\top A) \tag{6.33}$$

$$\text{with } A = rgrad(f) - \big(x\ symm(x\ rgrad(f)\big)$$

with *rgrad* being the Riemannian gradient as per eq. (6.21).

6.2.4 Consecutive Optimization on Rotations and Translations

The gradients and Hessians presented in this section (i.e., Section 6.2) up until this point regard the cost function in eq. (6.4), rewritten in the form of eq. (6.6). As mentioned, the bi-linearity of the cost function is tackled by solving for rotation (i.e., eq. (6.6)) and using the $R*$ solution obtained only after, when solving for translation (i.e., eq. (6.7)). The formulas for (Euclidean and Riemannian) gradients and Hessians have been found also for this second step of the algorithm, but are of more trivial derivation and are not explicitly reported in this dissertation. A big difference that makes the computation of the gradient and Hessians of eq. (6.7) easier lies in the fact that, being variable T in Euclidean space $\mathbb{R}^{d\times N}$, the Euclidean gradients and Hessians correspond with the Riemannian ones.

6.3 Riemannian Staircase

As briefly mentioned in the discussion of Chapter 2, Rosen *et al.* introduced the Riemannian Staircase (acronym RS) in the SE-Sync paper [86] to improve the computation of the optimal solution search in rank-restricted relaxation problems involving estimation of rotations and poses. Their goal is to find a set $Y \in St(d,m)^n$ of n Stiefel elements that minimizes the

$$\min_{Y \in St(d,m)^n} tr(f(Y^\top Y)) \tag{6.34}$$

for a certain linear map $f()$. In SE-Sync, the RS iteratively increases m and optimizes in spaces of dimension $d+1 \leq m \leq n$ to find better solutions than the locally optimal solutions obtained in previous steps of the staircase.

Unfortunately, the cost function of our problem (see eq. 6.17) cannot be easily mapped in the form of eq. 6.34. Hence, the need of a variation of SE-Sync's RS, where a different representation is adopted for the sets of rotations and translation, becomes apparent. This variation is presented in the following and is used to improve rotation and translation estimation accuracy of the cross-localization problem. Said variation consists in adding to each step of the staircase a translation estimation. As such, the original cost equations become (from eqs. (6.6), (6.7)), respectively:

1.
$$\min_{R \in St(d,nrs)^N} tr(R^\top L(T)R) + tr(R^\top P(T)) \tag{6.35}$$

2.
$$\min_{T \in \mathbb{R}^{nrs \times N}} ||A(R)vec(T) + b(R)||^2 \tag{6.36}$$

At each step the rotations lie on the Stiefel manifold, with iteratively increasing nrs, starting with $nrs = d$. The dimensions of the R_i, T_i members of cost function (recall eq. (6.4)) are set according to current step number of the staircase. Matrices L, P, A, B only vary in their size, but the formulas to obtain them remain the same e.g., the already mentioned $L(T)_{ij} = (T_j T_j^\top - T_i T_j^\top - T_j T_i^\top + T_i T_i^\top)$ block will just have

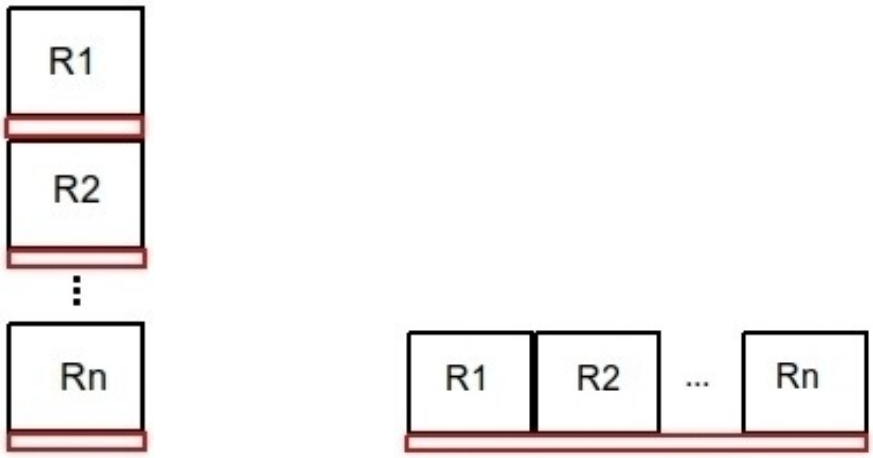


Figure 6.2: Riemannian Staircase paddings: Shape of Motion (left) VS SE-Sync [86] (right). The R_i blocks are rotation matrices, while the red rectangles indicate one padding row entirely composed of zeros. One zero-row is introduced at each new RS step.

$nrs \times d$ size instead of the initial $d \times d$ and composes block matrix L (now with size $(nrs\,N) \times (nrs\,N)$). The goal is to keep the cost equal to the one at previous step, after padding the R_i, T_i elements with zeros in the newly introduced dimension. An important difference with regard to the original formulation in [86] is that the $matStack()$ operator on the Stiefel 3D matrices stacks them vertically, while in the SE-Sync paper they are instead stacked horizontally. Another relevant difference w.r.t. SE-Sync's RS is that the dimension increase is done in a different fashion, as illustrated in Figure 6.2.

6.3.1 Computing New Descent Direction

As mentioned at the beginning of this chapter, the Riemannian Staircase is based on iteratively increasing the dimension of each rotation that is among the variables, starting from $SO(d) \subset St(d,d)$ and proceeding with $St(d,d+1), St(d,d+2)$ and so on. After running a first iteration of Riemannian optimization using the Rieman-

nian Trust-Regions (RTR) algorithm [125], the goal is to improve find a solution with lower cost than the first solution obtained x_0, while keeping a coherent formulation of the cost function. As a side note, RTR optimization is executed using the Manopt library [126]. In particular, the R, T_i, T_j vectors/matrices are expanded by one (or more) dimensions during the subsequent iterations (steps) of the staircase. The dimension augmentation is done by padding each R_i, T_i with zeros, without modifying the original associated cost of equation (6.4). Also, it can be shown that, if x_0 is a critical point for the initial cost function, then $x_{0\ padded}$ is also a critical point for the augmented-dimension cost function. Now, in order to actually exploit the subsequent steps of the staircase, it is necessary to compute a new starting descent direction in order to exit the (possibly) local minimum obtained at previous step. As said, the Staircase begins after the first iteration or RTR optimization, which outputs a matrix $x_0 \in St(d,d)^N$. The first goal becomes finding the minimum eigenvalue and associated eigenvector for the Hessian (linear) map $H(\cdot)$ applied on a point $x_0 \in St(d,p)^m$. In particular, let u be a vector tangent to x_0; consider $H(x_0)[u]$. The procedure to apply is presented in algs. 4, 5. Alg. 4 illustrates the algorithm used to search a negative eigenvalue and an associated eigenvector of the Riemannian Hessian (recall eq. 6.33). Said algorithm is based on a generalization of Von Mises' power iteration method [132]. The metric used for $x \in St(d,p)^m, v_1, v_2 \in \Diamond St(d,p)^m$ is as follows:

$$< x, v_1, v_2 >_{stief\_eucl} = multitrace(multiprod(multitransp(v_1), v_2)). \tag{6.37}$$

The operator defined in eq. (6.37) is called *Euclidean Stiefel metric*, and is used in alg. 4. The $\Diamond$ symbol indicates "tangent space".

Alg. 5 describes the general procedure for computing the solution of problems reported in eqs. (6.6)-(6.7) using RS. It starts by computing the highest norm eigenvalue through alg. 4. If this returns a negative eigenvalue $\lambda* < 0$ such that $\lambda * v* = H(x)[v*]$, then the search for a new starting point based on a line-search towards the direction of the associated eigenvector $v*$. Otherwise, a shift of the eigenvalues of the Hessian is performed by re-running P.I.M. on $H' = H - \mu v*$ in order to hopefully find a negative eigenvalue (of the original Hessian), and then perform the line-search. If no negative eigenvalue is found, the Riemannian Staircase procedure ends. Oth-

erwise, another instance of RTR optimization is called, this time on $St(d,d+1)^N$, starting from the newly found initial guess (computed through line-search).

Algorithm 4 Power Iteration Method applied to Hessian map on Stiefel manifold

Inputs: $H(\cdot), x \in St(d,p)^m$

Outputs: $[\lambda_{|max|}, v*] = P.I.M.(H(x)[u])$

$v \in \Diamond St(d,p)^m$

while Iterative outputs change $>$ threshold **do**

$v \leftarrow \frac{v}{<x,v,v>_{stief\_eucl}}$

$v \leftarrow -H(x)[v]$

end while

$v* = v$

$\lambda\_max = \frac{<v*,H(x)[v*]>_{stief\_eucl}}{<v*,v*>_{stief\_eucl}}$

6.4 Pose Averaging and Cross-Localization through SOM Experiments

For the CL-SOM project, experimental results regard preliminary tests on simulated benchmark sensor networks. In particular, a randomly generated network G with $N = 14$ nodes and a number of $45 \div 55$ edges has been used. On this generated benchmark, there is the possibility of setting the σ parameter that regulates how noisy the $R_i, T_i, T_i j$ members of cost function (see eq. (6.4)) are. The MATLAB implementation used during the experiments is publicly available online[1]. On the described benchmark network, the following scenario has been tested. The CL-SOM algorithm is run multiple times (in general 50 times) with the same noise level, in order to gather a clearer picture of the robustness of each algorithm. The assessed metrics per each σ are mean rotation error, mean translation error and mean execution time. In particular, rotation error per each CL-SOM run in each sigma is computed as the geodesic distance [130, 133] between the obtained rotations and the ground-truth generated in

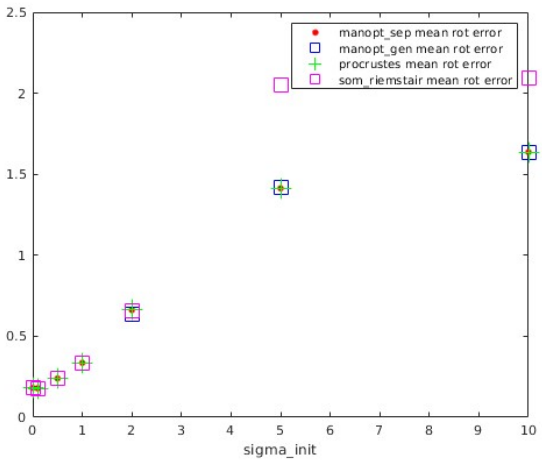
[1] https://github.com/ErnestF22/som/

Algorithm 5 Get minimum eigenvalue and associated eigenvector of Hessian map on Stiefel manifold

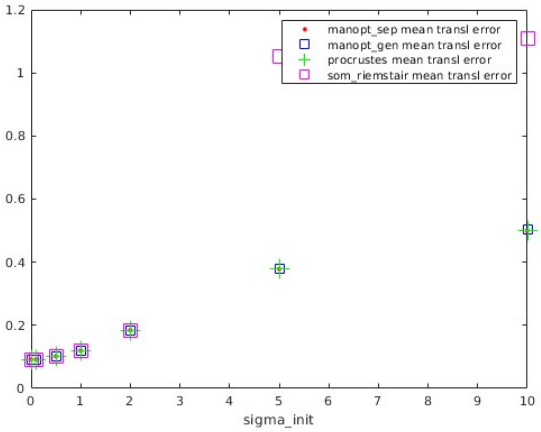
$[\lambda_{|max|}(H), u*] \leftarrow P.I.M.(H(x)[u])$ // see alg. 4
if $\lambda_{|max|}(H) < 0$ **then**
  $\lambda_{min} = \lambda_{|max|}(H)$
else
  $H' = H - \mu u*$ with $\mu \geq \lambda_{|max|}(H)$
  $[\lambda_{|max|}(H'), u*] \leftarrow P.I.M.(H'(x)[u])$
  $\lambda_{min} = \lambda_{|max|}(H') + \mu$
end if
if $\lambda_{min} < 0$ **then**
  $new\_descent\_dir = linesearch(x, \lambda_{min}, u*)$
  $manopt_{stief}()$ // new Riemannian staircase step
else
  End of Riemannian staircase
end if

the benchmark. Translation error is more easily the norm of the difference between estimated translations and ground truth.

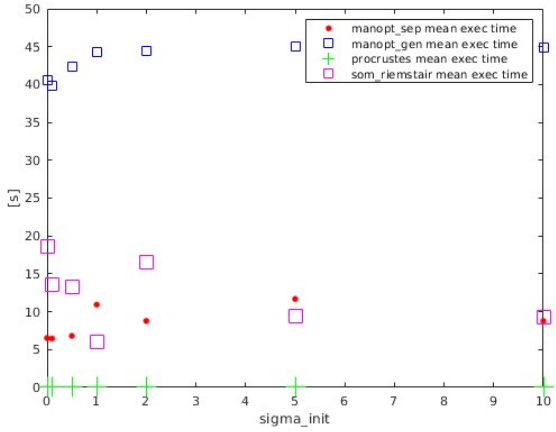
The presented CL-SOM (labeled *som_riemstair* in Figures 6.3, 6.4) algorithm has been tested against three other methods. The first method (labeled *manopt_sep*) basically consists in running just the first iteration of the RS, with $nrs = d$, and stop there. The first method (labeled *manopt_gen*) is based on the same functioning concepts as *manopt_sep*, but makes use of the *generalized_procrustes* implementation that can be found in the Manopt library [125]. This is a procedure designed and implemented in order to optimize the resolution of this type of problem, by avoiding separate iterations for rotation and translation (e.g., solve (6.6) and then (6.7)) and instead optimize simultaneously for both of them. The third comparison method (labeled *procrustes*) is a variation of the Procrustes solution through SVD [92, 134, 135], inspired by the Kabsch-Umeyama algorithm [136, 137]. For all metrics, and all methods, a mean of all results obtained for a certain σ is performed. To give a better understanding of why these have been the chosen comparison methods, it can help knowing that, with limited noise the first two methods should converge (through retractions based on Riemannian optimization theory) to the same solution that the third method outputs through matricial operations. The goal of CL-SOM is to increase accuracy in the pose estimation through the subsequent steps of RS, and consequently improve *manopt_sep* and *manopt_gen* results. The results for σ varying in set $\sigma = \{0.0, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0\}$ are reported in Figure 6.3. These σ values are the variance of the Gaussian noise added to the ground truth information, for obtaining the initial guesses of the $\{R_i, T_i\}$ transformations. Taking a look at said Figure, it can be seen that CL-SOM (labeled *som_riemstair*) presents a mean rotation error on par with the one of the other methods with the lower end of noise values. Instead, while the translation errors are very similar across all methods, CL-SOM has higher execution times than the ones of *manopt_sep* and *procrustes*. The limited improvement can be due to the limited quality of the eigenvalue-eigenvector couples extracted through algs. 4, 5, especially in the more noisy cases. Also, the quality of the RS results needs to be improved in the tests with higher σ, so more work on the stability of numerical computations performed when adding steps to the RS needs



(a) Rotation Mean Errors.



(b) Translation Mean Errors.



(c) Mean Execution Times.

Figure 6.3: CL-SOM mean rotation error, translation error and execution time results on randomly generated benchmark with $N = 14$ nodes.

to be done. Hence, an alternative version of the test with slightly different setup has been proposed and run.

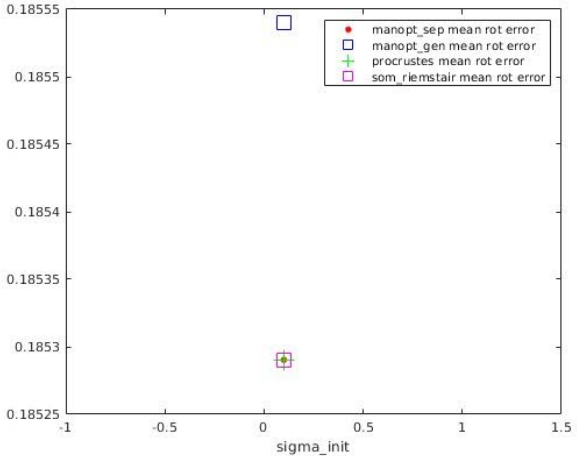
It has to be noted that optimization methods using Riemannian Geometry solvers rely on the quality of initial guess, which can deeply affect accuracy and speed of the algorithms. As such, in order to simulate a fleet of sensors that iteratively gathers information on the reciprocal pose of the other sensors, the initial guess for the N nodes has been constructed starting by a randomly selected couple of sensors, running the algorithms on this sub, and add the remaining nodes one by one only afterwards. The initial guess of the very first iteration, as well as the initial guess for the newly added node are generated randomly.

After obtaining all N poses $\{R_i, T_i\}$, the algorithms are run one last time on the full set of nodes. The results for this test with $\sigma = 0.1$ (used for adding noise to the T_{ij} edge values) are reported in Figure 6.4.

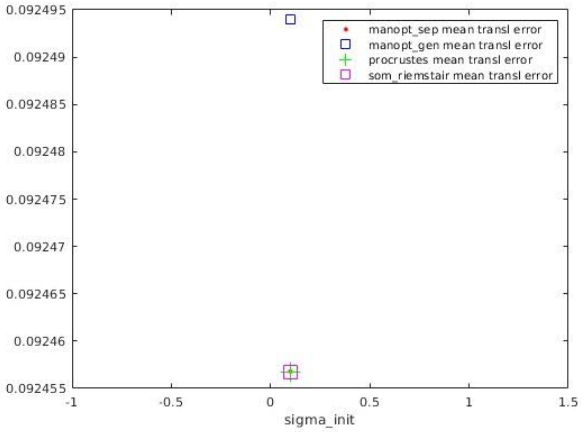
Here, small improvements for CL-SOM (on the 10^{-6} order) can be seen, even in the most relevant case of reducing rotation errors. Still, this comes at a noticeable computational cost increase compared to both *procrustes* and *manopt_sep*.

6.5 Discussion

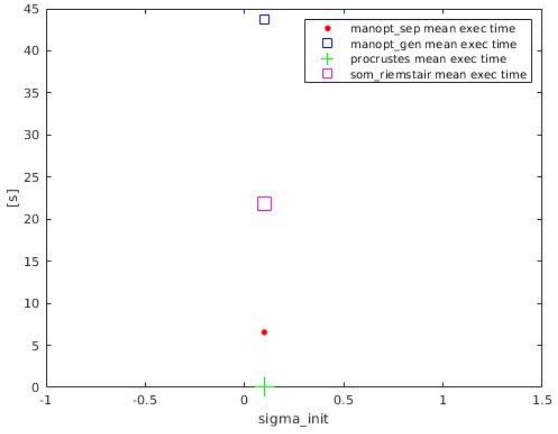
This chapter has presented a method based on the Riemannian Staircase paradigm, which aims to solve the cross-localization of a fleet of sensors (e.g., drones). The discussed approach is based on Riemannian Geometry and its related optimization theory, and has the goal of improving existing methods, especially in the absence of initial guess for the poses of the fleet. To sum up experimental tests and results, more relevant improvements are needed to fully justify the added computational cost that is naturally introduced by the RS addition, but encouraging results have been found even with relatively limited noise. In future works, additional extensive testing (involving not only simulated benchmarks) will be done in order to thoroughly prove the effectiveness of the RS addition, also in real experimental setups, with the goal of assessing real advantages over state-of-the-art methods, with the goal of bringing the qualitative improvements with a more limited computational complexity.



(a) Rotation Mean Errors.



(b) Translation Mean Errors.



(c) Rotation Mean Errors.

Figure 6.4: CL-SOM results with $\sigma = 0.1$ and random initial guess.

Chapter 7

Conclusion

This dissertation has addressed robust geometric methods used in estimation problems in robotics. In particular, the focus has been on point cloud registration, which is an important primitive in many simultaneous localization and mapping systems, and on multi-agent or multi-sensor cross localization. The approach followed in this work is grounded on geometric models and on the manifold structure of the variables involved in estimation, in order to achieve global and robust solutions not relying on initial guess. The dissertation has opened with an analysis of state of the art methods in these fields, explaining the reasoning behind the selected approaches. Then, two main contributions have been presented.

The first one regards the Angular Radon Spectrum (ARS) formulation, in the context of pose estimation and, more in general, SLAM. Angular Radon Spectrum is a descriptor for Gaussian Mixture Models (GMMs), which is able to capture the collinearity (or coplanarity in 3D) of pencils of lines (or planes) that compose the GMM. This allows decoupling of rotation estimation from translation estimation. Due to this, ARS sees its most relevant use case in the ability to estimate the rotation separating a couple of partially overlapping point clouds, through maximization of the correlation between the two spectra. Hence, the main contributions regarding ARS have been the Anisotropic version for the planar case, the GPU-based parallelization of the planar isotropic version, and the 3D isotropic version. The ways of converting

a sensory point cloud into an isotropic or generic GMM have also been discussed. To complete pose estimation, a branch-and-bound algorithm for translation has been proposed. Pose estimation through ARS is able to achieve accurate results with computational speed comparable to other state of the art methods, as discussed during the experimental evaluation.

The second main contribution presented is the cross-localization of a sensor fleet through a formulation based on the Shape of Motion matrix. The novelty introduced by this approach regards the reformulation of the Riemannian Staircase (RS) algorithm to the specific case needed. RS is used to exit local minima in which solvers stop with a given dimensionality of the problem, by iteratively increasing the dimension of the cost function members and their associated search space. Optimization has been performed through Riemannian Geometry and Netwonian solvers. A relevant part of the work regards the search of an exit direction leading to a point with cost lower than the one where the optimization stopped at the previous staircase step, based on searching a negative eigenvalue of the Riemannian Hessian of the cost function, after increasing the dimensionality of the problem.

Future works on the ARS project will aim at improving the optimization procedure in the 3D case, while limiting the computational complexity of the algorithm. The goal is the one of obtaining a full and efficient SLAM pipeline, whose core relies on the accuracy in point cloud alignment of the ARS formulation. For what concerns the cross-localization through SOM project, the goal is to test the full implementation on a fleet of drones, after obtaining clearer improvements over other comparable methods, especially in the absence of initial guess.

Appendix A

Math Appendix for ARS

1.1 Connection Coefficients from Legendre to Tchebychev Polynomials

Tchebychev and Legendre polynomials are special cases of *ultraspherical* or *Gegenbauer polynomials*, which in turn are a special case of *Jacobi polynomials*. The identification of a general definition for different kinf of polynomials enables to write one kind of polynomial as a linear combination of another kind of polynomial. In particular, the results [113, Theorem 9.1.1] and, more specifically, [113, Theorem 9.1.2] provide the *connection coefficients* between Gegenbauer polynomials

$$C_N^{\gamma}(x) = \sum_{k=0}^{\lfloor N/2 \rfloor} \frac{(\gamma-\beta)_k \, (\gamma)_{N-k} \, (\beta+N-2k)}{k!(\beta+1)_{N-k} \, \beta} \, C_{N-2k}^{\beta}(x) \tag{A.1}$$

where $C_N^{\gamma}(x)$ and $C_N^{\beta}(x)$ are two Gegenbauer polynomials respectively with parameter γ and β. If N is even, $C_N^{\gamma}(x)$ depends on $C_l^{\beta}(x)$ with even order $l = 0, 2, \ldots, N$. If N is odd, $C_N^{\gamma}(x)$ depends on $C_l^{\beta}(x)$ with odd order $l = 1, 3, \ldots, N$. The above equation uses the Pochammer symbols as a notation for *rising factorial power* that is defined for a generic $v \in \mathbb{R}$ and integer $n \leqslant 0$ as

$$(v)_n = v(v+1)\ldots(v+n-1) \tag{A.2}$$

Note that the rising factorial power in the special case $v = 1$ becomes the well known factorial $(1)_n = n!$. There are alternative notations to Pochammer symbols, notably $v^{\bar{n}}$ in [138]. The Legendre polynomials are ultraspherical polynomials corresponding to $\nu = 1/2$, $P_N(x) = C_N^{1/2}(x)$. The Tchebychev polynomials corresponds to $\beta = 0$ that can be set only as a limit [113, Eq. (4.5.22), p. 97]

$$T_N(x) = \lim_{\nu \to 0} \frac{N+2\nu}{2\nu} C_N^{\nu}(x) \tag{A.3}$$

There are equivalent limits that can be used to derive the formula refering to slightly different connection. In [139] the limit procedure for finding the connection coefficients is

$$T_N(x) = \lim_{\nu \to 0} \frac{\Gamma(N+1)}{(2\nu)_N} C_N^{\nu}(x) \tag{A.4}$$

The computation of coefficients in eq. (A.1) for the specific case needed is as follows:

$$\begin{aligned}
\hat{\ell}_{N,k} &= \frac{\Gamma(N+1)}{(2\gamma)_N} \frac{(\gamma-1/2)_k \, (\gamma)_{N-k} \, (N-2k+1/2)}{k! \, (1/2+1)_{N-k} \, (1/2)} \\
&= \underbrace{\frac{(\gamma-1/2)_k (\gamma)_{N-k}}{(2\gamma)_N}}_{\xi(\gamma)} \frac{N! \, (N-2k+1/2)}{k! \, (1/2)_{N-k+1}} \\
&= \frac{\left(-\frac{1}{2}\right)_k (N-k-1)!}{2} \frac{N(N-2k+\frac{1}{2})}{k! \, \left(\frac{1}{2}\right)_{N-k+1}}
\end{aligned}$$

where the term $\xi(\gamma)$ related to the parameter γ, $\Gamma(N+1) = N!$ has been substituted and also

$$\begin{aligned}
&(1/2)(1/2+1)_{N-k} \\
&\quad = \left(\frac{1}{2}\right) \left(\frac{1}{2}+1\right) \left(\frac{1}{2}+2\right) \ldots \left(\frac{1}{2}+1+N-k-1\right) \\
&\quad = \left(\frac{1}{2}\right) \left(\frac{1}{2}+1\right) \ldots \left(\frac{1}{2}+N-k\right) = \left(\frac{1}{2}\right)_{N-k+1}
\end{aligned} \tag{A.5}$$

Observe that the integer index $k = 0, \ldots, \lfloor N/2 \rfloor$ and $k \leqslant N$ (equality when $N = 0$). Now, it is important to investigate the value of $\xi(\gamma)$ when γ approach 0.

$$\begin{aligned}
\xi(\gamma) &= \frac{(\gamma - 1/2)_k \ (\gamma)_{N-k}}{(2\gamma)_N} \\
&= \frac{(\gamma - 1/2)_k \cdot \gamma \ (\gamma+1) \ldots (\gamma+N-k-1)}{2\gamma \ (2\gamma+1) \ldots (2\gamma+N-1)} \\
&= \frac{(\gamma - 1/2)_k \cdot (\gamma+1) \ldots (\gamma+N-k-1)}{2 \ (2\gamma+1) \ldots (2\gamma+N-1)} \\
&= \frac{(\gamma - 1/2)_k \ (\gamma+1)_{N-k-1}}{2 \ (2\gamma+1)_{N-1}}
\end{aligned} \tag{A.6}$$

Thus, the limit for $\gamma \to 0$ leads to

$$\lim_{\gamma \to 0} \xi(\gamma) = \frac{(-1/2)_k \ (N-k-1)!}{2 \ (N-1)!} \tag{A.7}$$

Setting $N = 2$, it follows that

$$T_2(x) = \sum_{k=0}^{\lfloor \frac{N}{2} \rfloor = 1} \hat{l}_{N,k} * P_{N-2k}(x) \tag{A.8}$$

from which $k = 0, 1$. Expanding the corresponding expressions, the following comes out:

$$l_{2,0} = \frac{4}{3} \tag{A.9}$$

$$l_{2,1} = -\frac{1}{3} \tag{A.10}$$

which are the same results found in: [140]

Finally, the expression of the connection coefficients can be inferred:

$$\ell_{N,k} = \frac{\left(-\frac{1}{2}\right)_k (N-k-1)!}{2} \ \frac{N(N-2k+\frac{1}{2})}{k! \ \left(\frac{1}{2}\right)_{N-k+1}} \tag{A.11}$$

$$= \underbrace{\frac{\left(-\frac{1}{2}\right)_k}{k!}}_{a_\ell(k)} \underbrace{\frac{N \ (N-2k+\frac{1}{2}) \ (N-k-1)!}{2 \ \left(\frac{1}{2}\right)_{N-k+1}}}_{b_\ell(k)} \tag{A.12}$$

The expression has been decomposed into two terms $a_\ell(k)$ and $b_\ell(k)$ to derive the recursion in k when $k > 0$:

$$\begin{aligned}
\frac{a_\ell(k)}{a_\ell(k-1)} &= \frac{\left(-\frac{1}{2}\right)_k}{k!}\,\frac{(k-1)!}{\left(-\frac{1}{2}\right)_{k-1}} = \frac{\left(-\frac{1}{2}+k-1\right)\left(-\frac{1}{2}\right)_{k-1}}{\left(-\frac{1}{2}\right)_{k-1}}\,\frac{1}{k} \\
&= \frac{2k-3}{2k} \\
\frac{b_\ell(k)}{b_\ell(k-1)} &= \frac{N-2k+1/2}{N-2(k-1)+1/2}\,\frac{(N-k-1)!}{(N-k)(N-k-1)!}\cdot \\
&\quad \cdot \frac{\left(\frac{1}{2}\right)_{N-k+1}\left(\frac{1}{2}+N-k+1\right)}{\left(\frac{1}{2}\right)_{N-k+1}} \\
&= \frac{N-2k+1/2}{N-2k+5/2}\,\frac{N-k+3/2}{(N-k)}
\end{aligned}$$

The next step is the derivation of a recurrence relation with respect to both N and k. First, the connection coefficients when $k = 0$ are

$$\ell_{0,0} = 1 \tag{A.13}$$

$$\begin{aligned}
\ell_{N,0} &= \frac{\left(-\frac{1}{2}\right)_0\, N!\,\left(N+\frac{1}{2}\right)}{2\;0!\;\left(\frac{1}{2}\right)_{N+1}} = \frac{N!\,\left(N+\frac{1}{2}\right)}{2\,\frac{1\cdot 3\cdot\ldots\cdot(2N+1)}{2^{N+1}}} \\
&= \frac{N+\frac{1}{2}}{2}\,\frac{2^{N+1}N!}{(2N+1)!!} = \left(N+\frac{1}{2}\right)\,\frac{2^N N!}{(2N+1)!!} \\
&= \left(N+\frac{1}{2}\right)\,\frac{(2N)!!}{(2N+1)!!} \qquad \text{(A.14)} \\
&= \left(N+\frac{1}{2}\right)\,\frac{2N}{2N+1}\,\frac{2(N-1)}{2(N-1)+1}\cdots\frac{2\,2}{2\,2+1}\,\frac{2\,1}{2\,1+1} \qquad \text{(A.15)}
\end{aligned}$$

The above eq. (A.14) holds for $N > 0$. The reader can check that direct substitution of $N = 0$ yields $1/2$ instead of $\ell_{0,0}$. If $N \geqslant 2$, then the eq. A.14 holds for both $\ell_{N,0}$ and $\ell_{N,0}$ and there is the recurrence relation

$$\ell_{N,0} = \ell_{N-1,0}\,\frac{N+\frac{1}{2}}{N-\frac{1}{2}}\,\frac{2N}{2N+1} \tag{A.16}$$

Finally, the recurrence relations of $a_\ell(k)$ and $b_\ell(k)$ are used to derive the recurrence

relation on k as

$$\ell_{N,k} = \ell_{N,k-1} \, \frac{2k-3}{2k} \, \frac{N-2k+1/2}{N-2k+5/2} \, \frac{N-k+3/2}{N-k} \tag{A.17}$$

In summary, the table of connection coefficients can be computed by executing the following step.

1. Set $\ell_{0,0} = \ell_{1,0} = 1$ and $N = 2$.
2. Compute $\ell_{N,0}$ from $\ell_{N-1,0}$ using eq. A.16. .
3. Compute $\ell_{N,k}$ from $\ell_{N,k-1}$ for $k = 1, 2, \ldots, \lfloor \frac{N}{2} \rfloor$ using eq. (A.17) (in increasing order of k to have the term $k-1$).
4. If $N < N_{max}$ (N_{max} is the required maximum order of the Tchebychev polynomial to be computed), increase $N \leftarrow N+1$ and go to step 2.

1.2 Spherical Harmonics: Useful Formulas

1.2.1 Complex and Real Spherical Harmonics

Let S^2 be the ordinary sphere consisting of the points with distance 1 from the origin in $\mathbb{R}^3$. The following common notation is used:

1. $Y_l^m(\theta, \varphi)$ for the complex SH or CSH: $S^2 \to \mathbb{C}$ defined as

$$Y_l^m(\theta, \varphi) = \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} \, P_l^m(\cos\theta) \, \mathrm{e}^{im\varphi} \tag{A.18}$$

   where the alternating sign term $(-1)^m$ is encapsulated by associated Legendre polynomial P_l^m.

2. $Y_{lm}(\theta, \varphi)$ for the real-valued SH or RSH: $S^2 \to \mathbb{R}$, also called at times tesseral

SH, is defined as

$$Y_{lm}(\theta,\varphi) = \begin{cases} (-1)^m\sqrt{2}\sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}\,P_l^{|m|}(\cos\theta)\,\sin(|m|\varphi) & m<0 \\ \sqrt{\frac{2l+1}{4\pi}}\,P_l^0(\cos\theta) & m=0 \\ (-1)^m\sqrt{2}\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\,P_l^m(\cos\theta)\,\cos(m\varphi) & m>0 \end{cases} \tag{A.19}$$

1.2.2 Conversion Formulas between CSH and RSH

CSH and RSH are related by the following equations. To convert CSH into RSH there is the equation

$$Y_l^m(\theta,\varphi) = \begin{cases} \frac{1}{\sqrt{2}}Y_{l,|m|}(\theta,\varphi) - \frac{i}{\sqrt{2}}Y_{l,-|m|}(\theta,\varphi) & m<0 \\ Y_{l0}(\theta,\varphi) & m=0 \\ \frac{(-1)^m}{\sqrt{2}}Y_{l,|m|}(\theta,\varphi) + \frac{i(-1)^m}{\sqrt{2}}Y_{l,-|m|}(\theta,\varphi) & m>0 \end{cases} \tag{A.20}$$

Conversely, the conversion from RSH to CSH is given by

$$Y_{l,m}(\theta,\varphi) = \begin{cases} \frac{i}{\sqrt{2}}Y_l^m(\theta,\varphi) - \frac{(-1)^m i}{\sqrt{2}}Y_l^{-m}(\theta,\varphi) & m<0 \\ Y_l^0(\theta,\varphi) & m=0 \\ \frac{(-1)^m}{\sqrt{2}}Y_l^m(\theta,\varphi) + \frac{1}{\sqrt{2}}Y_l^{-m}(\theta,\varphi) & m>0 \end{cases} \tag{A.21}$$

1.3 Gradient of ARS3D correlation function

1.3.1 Rotations around Z-axis

Let's consider a rotation of an angle γ around the Z-axis. The Real Spherical Harmonics (RSH) matrices [115] associated to such rotation are as follows:

$$l = 0 \rightarrow M_0 = 1 \tag{A.22}$$

$$l = 1 \rightarrow M_1 = \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \tag{A.23}$$

$$l = 2 \rightarrow M_2 = \begin{bmatrix} \cos(\gamma)^2 - \sin(\gamma)^2 & 0 & 0 & 0 & 2\cos(\gamma)\sin(\gamma) \\ 0 & \cos(\gamma) & 0 & \sin(\gamma) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -\sin(\gamma) & 0 & \cos(\gamma) & 0 \\ -2\cos(\gamma)\sin(\gamma) & 0 & 0 & 0 & \cos(\gamma)^2 - \sin(\gamma)^2 \end{bmatrix} \tag{A.24}$$

$$l = 3 \rightarrow M_3 = \begin{bmatrix} \cos(\gamma)(\cos(\gamma)^2 - \sin(\gamma)^2) - 2\cos(\gamma)\sin(\gamma)^2 & 0 \cdots 0 & \sin(\gamma) * (\cos(\gamma)^2 - \sin(\gamma)^2) + 2\cos(\gamma)^2\sin(\gamma) \\ \vdots & M_2 & \vdots \\ -\sin(\gamma)(\cos(\gamma)^2 - \sin(\gamma)^2) - 2\cos(\gamma)^2\sin(\gamma) & 0 \cdots 0 & \cos(\gamma)(\cos(\gamma)^2 - \sin(\gamma)^2) - 2\cos(\gamma)\sin(\gamma)^2 \end{bmatrix} \tag{A.25}$$

$$l = 4 \rightarrow M_4 = \ldots$$

Note that the *vdots* in M_3 are in fact zeros. It appears clear that M_{l+1} has M_l in the center, and then 4 corner elements on the borders of the matrix, with zeros in-between (both vertically and horizontally). Let the element in the upper leftmost corner be called a_{l+1}, and the element in the upper rightmost corner b_{l+1}. Then, we know that other elements are replicated +/- a sign. So, the recursive formula for M_{l+1} is as

follows:

$$M_{l+1} = \begin{bmatrix} a_{l+1} & \cdots & b_{l+1} \\ \vdots & M_l & \vdots \\ -b_{l+1} & \cdots & a_{l+1} \end{bmatrix} \tag{A.26}$$

where

$$a_{l+1} = \cos\gamma\, a_l - \sin\gamma\, b_l \tag{A.27}$$

$$b_{l+1} = \sin\gamma\, a_l + \cos\gamma\, b_l \tag{A.28}$$

It can be observed that, since $a_1 = \cos\gamma$ and $b_1 = \sin\gamma$, it is apparent that the general form of the two variables is

$$a_{l+1} = \cos(l\gamma) \tag{A.29}$$

$$b_{l+1} = \sin(l\gamma) \tag{A.30}$$

Hence, the general form of the RSH matrices is the following:

$$\mathbf{M}_l^z(\gamma) = \mathbf{M}_l(\mathbf{R}_z(\gamma)) = \begin{bmatrix} \cos(l\gamma) & \cdots & 0 & 0 & 0 & 0 & \sin(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & \cos\gamma & 0 & \sin\gamma & & 0 \\ 0 & & 0 & 1 & 0 & & 0 \\ 0 & & -\sin\gamma & 0 & \cos\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ -\sin(l\gamma) & 0 & 0 & 0 & 0 & \cdots & \cos(l\gamma) \end{bmatrix} \tag{A.31}$$

By expanding the sines and cosines in $\mathbf{M}_l^z(\gamma)$ as

$$\cos(m\gamma) = \sum_{k\geqslant 0} \frac{(m\gamma)^{2k}}{(2k)!} \tag{A.32}$$

$$\sin(m\gamma) = \sum_{k\geqslant 0} \frac{(-1)^k (m\gamma)^{2k+1}}{(2k+1)!} \tag{A.33}$$

the power terms can be collected as

$$\mathbf{M}_l^z(\gamma) = \exp\left(\underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & -l\gamma \\ 0 & 0 & \cdots & -(l-1)\gamma & 0 \\ \vdots & & & & \vdots \\ \vdots & (l-1)\gamma & \cdots & 0 & 0 \\ l\gamma & 0 & \cdots & 0 & 0 \end{bmatrix}}_{\mathbf{L}_l^z(\gamma)}\right) \tag{A.34}$$

where $L_l^z(\gamma)$ is the Lie algebra associated to $\mathbf{M}_l^z(\gamma)$.

The matrix of the derivatives and twice derivatives of the terms of $\mathbf{M}_l^z(\gamma)$ w.r.t. γ are

$$\dot{\mathbf{M}}_l^z(\gamma) = \begin{bmatrix} -l\sin(l\gamma) & \cdots & 0 & 0 & 0 & 0 & l\cos(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & -\sin\gamma & 0 & \cos\gamma & & 0 \\ 0 & & 0 & 0 & 0 & & 0 \\ 0 & & -\cos\gamma & 0 & -\sin\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ -l\cos(l\gamma) & 0 & 0 & 0 & 0 & \cdots & -l\sin(l\gamma) \end{bmatrix} \tag{A.35}$$

$$\ddot{\mathbf{M}}_l^z(\gamma) = \begin{bmatrix} -l^2\cos(l\gamma) & \cdots & 0 & 0 & 0 & 0 & -l^2\sin(l\gamma) \\ \vdots & \ddots & & & & & \vdots \\ 0 & & -\cos\gamma & 0 & -\sin\gamma & & 0 \\ 0 & & 0 & 0 & 0 & & 0 \\ 0 & & \cos\gamma & 0 & -\cos\gamma & & 0 \\ \vdots & & & & & \ddots & \vdots \\ l^2\sin(l\gamma) & 0 & 0 & 0 & 0 & \cdots & -l^2\cos(l\gamma) \end{bmatrix} \tag{A.36}$$

1.3.2 Rotations around Y-axis

Let's consider a rotation of an angle β around the Y-axis. The Real Spherical Harmonics (RSH) matrices associated to such rotation are as follows:

$$l = 0 \rightarrow M_0 = 1 \tag{A.37}$$

$$l = 1 \rightarrow M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \tag{A.38}$$

$$l = 2 \rightarrow M_2 = \begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 & 0 & 0 \\ -\sin(\beta) & \cos(\beta) & 0 & 0 & 0 \\ 0 & 0 & \cos(\beta)^2 - \frac{\sin(\beta)^2}{2} & -\sqrt{3}\cos(\beta)\sin(\beta) & \frac{\sqrt{3}\sin(\beta)^2}{3} - \frac{\sqrt{3}(\cos(\beta)^2-1)}{6} \\ 0 & 0 & \sqrt{3}\cos(\beta)\sin(\beta) & \cos(\beta)^2 - \sin(\beta)^2 & -\cos(\beta)\sin(\beta) \\ 0 & 0 & \frac{\sqrt{3}\sin(\beta)^2}{2} & \cos(\beta)\sin(\beta) & \frac{\cos(\beta)^2}{2} + \frac{1}{2} \end{bmatrix} \tag{A.39}$$

$$l = 3 \rightarrow M_3 = \ldots \tag{A.40}$$

$$l = 4 \rightarrow M_4 = \ldots \quad .$$

It is apparent that M_l quickly becomes hard to use. Due to this, the following approach has been taken: consider a permutation matrix E s.t. $R_Y(\beta) = E^\top R_Z(\beta) E$. We want to verify that, for

$$E = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.41}$$

the following is true:

$$M_l(R_Y(\beta)) = M_l(E^\top)\, M_l(R_Z(\beta))\, M_l(E) \tag{A.42}$$

so that the approach for Rotations around Z-axis described in Section 1.3.1 can be (at least partially) reused. An important note is that E can serve also as a rotation, and so its associated RSH matrices can be calculated and considered. Also, a similar procedure to the one just described can be adopted to tackle rotations around the X-axis, but it is not expanded on here, as the ZYZ Euler angles notation has been the one selected.

1.3.3 Euclidean Gradient through inverse Jacobian of Euler Angles associated matrix

To obtain the element-wise Euclidean gradient of the cost function, we start from expressing the Jacobian J of the function associated to matrix R_{zyz}. R_{zyz} is the rotation matrix that produces the rotation, corresponding to a given $[\alpha,\beta,\gamma]$ tuple of ZYZ Euler Angles [141]. First, let $M(\alpha)_l, M(\beta)_l, M(\gamma)_l$ be the RSH matrices of order l associated to α,β,γ angle rotations around Z-axis. Then, let $dM(\alpha)_l, dM(\beta)_l, dM(\gamma)_l$ be their respective coefficient-wise derivative. The Euclidean gradient of the cost function is obtained by multiplying the pseudo-inverse of J, call it J_{pinv}, by the $grad_{eul}$ 3×1 vector obtained as follows (using MATLAB array indexing):

$$grad_{eul}(1) = \sum_{l=1}^{l_{max}} rsh\_src_l^\top M(\gamma)_{l+1} M(E)_{l+1}^\top M(\beta)_{l+1} M(E)_{l+1} dM(\alpha)_{l+1} rsh\_dst_l \tag{A.43}$$

$$grad_{eul}(2) = \sum_{l=1}^{l_{max}} rsh\_src_l^\top M(\gamma)_{l+1} M(E)_{l+1}^\top dM(\beta)_{l+1} M(E)_{l+1} M(\alpha)_{l+1} rsh\_dst_l \tag{A.44}$$

$$grad_{eul}(3) = \sum_{l=1}^{l_{max}} rsh\_src_l^\top dM(\gamma)_{l+1} M(E)_{l+1}^\top M(\beta)_{l+1} M(E)_{l+1} M(\alpha)_{l+1} rsh\_dst_l \tag{A.45}$$

where E is the matrix from eq. (A.41), $M(E)$ its associated RSH matrices,

$$rsh\_src_l = rsh\_src(sh\_lm\_to\_index(l,-l) : sh\_lm\_to\_index(l,l)) \tag{A.46}$$

$$rsh\_dst_l = rsh\_dst(sh\_lm\_to\_index(l,-l) : sh\_lm\_to\_index(l,l)) \tag{A.47}$$

$$\text{where } \; sh\_lm\_to\_index(l,m) = l^2 + m + l + 1, \;\; \text{if} \;\; -l \le m \le l \tag{A.48}$$

and rsh_src, rsh_dst are the RSH coefficients of *source* and *destination* respectively. The already mentioned final step to obtain the Euclidean gradient $\overline{grad}$ is

$$\overline{grad} = grad_{eul} J_{pinv} \;\; . \tag{A.49}$$

At the end of the described procedure, $\overline{grad}$ from eq. (A.49) ends up being a 9-element array, that can be reshaped according to library needs e.g., most optimization libraries require it as a 3×3 matrix.

1.4 Lie Algebra of Rotation of Spherical Harmonics

Further expanding on some tools used in the previous sections of this appendix, in general, the Rotation Spherical Harmonics (RSH) matrices $\mathbf{M}^l(\mathbf{R}) \in SO(2l+1)$ are orthonormal matrices with an associated Lie algebra. Hence, it would be interesting to find for every $\mathbf{M}^l(\mathbf{R})$ its corresponding Lie algebra matrix $\mathbf{L}^l$. The RSH matrices $\mathbf{M}^l(\mathbf{R})$ have a simple analytical expression when the rotation $\mathbf{R}$ is a rotation around axis z. In the previous section, the following Lie algebra matrix has been obtained:

$$\mathbf{\Omega}_z^l(\gamma) = \begin{bmatrix} 0 & 0 & \cdots & 0 & -l\gamma \\ 0 & 0 & \cdots & -(l-1)\gamma & 0 \\ \vdots & & & & \vdots \\ \vdots & (l-1)\gamma & \cdots & 0 & 0 \\ l\gamma & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{A.50}$$

$$\mathbf{M}_l^z(\gamma) = \exp\left(\mathbf{L}_l^z(\gamma)\right) \tag{A.51}$$

The Lie algebra for a generic rotation $\mathbf{R} \in SO(3)$ can be obtained after rewriting the rotation matrix as follows. The rotation $\mathbf{R}$ can be represented using its axis $\mathbf{a}$ and rotation angle α. The axis is a unit vector that can be represented by polar coordinates $\mathbf{a} = [\sin\theta\cos\varphi,]\sin\theta\sin\varphi, \cos\theta]^\top$. Thus, the rotation can be written as follows:

$$\mathbf{R} = \mathbf{R}_z(\varphi)\mathbf{R}_y(\theta)\mathbf{R}_z(\alpha)\mathbf{R}_y^\top(\theta)\mathbf{R}_z^\top(\varphi) \tag{A.52}$$

For convenience we define the orthonormal matrix $\mathbf{U} = \mathbf{R}_z(\varphi)\mathbf{R}_y(\theta)$. The Lie algebra can be obtained using the following property:

$$\mathbf{M}^l(\mathbf{R}) = \mathbf{M}^l(\mathbf{U})\mathbf{M}^l(\mathbf{R}_z(\alpha))bM^{l\top}(\mathbf{U}) \tag{A.53}$$

$$= \mathbf{M}^l(\mathbf{U})\exp\left(\mathbf{\Omega}_z^l(\alpha)\right)bM^{l\top}(\mathbf{U}) \tag{A.54}$$

where the exponential of the Lie algebra element $\mathbf{\Omega}_z^l(\alpha)$ has been substituted. Also, the following property of matrix exponential is exploited:

$$\mathbf{M}^l(\mathbf{R}) = \exp\left(\mathbf{M}^l(\mathbf{U})\mathbf{\Omega}_z^l(\alpha)\mathbf{M}^{l\top}(\mathbf{U})\right) \tag{A.55}$$

Bibliography

[1] J. Sola, J. Deray, and D. Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[2] R.B. Rusu. 3d point feature representations. *Semantic 3D Object Maps for Everyday Robot Manipulation*, pages 33–60, 2013.

[3] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3212–3217, 2009.

[4] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827, 2007.

[5] J. Ma, J. Zhao, J. Tian, A.L. Yuille, and Z. Tu. Robust Point Matching via Vector Field Consensus. *IEEE Transactions on Image Processing*, 23(4):1706–1721, April 2014.

[6] D. Holz, A.E. Ichim, F. Tombari, R.B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, Dec 2015.

[7] P.J. Besl and H.D. Mckay. A method for registration of 3-d shapes. *IEEE Trans. Pat. Anal. Mach. Intel*, 14(2):239–256, 1992.

[8] J. Serafin and G. Grisetti. Using extended measurements and scene merging for efficient and robust point cloud registration. *RAS*, 92:91–106, 2017.

[9] E.B. Olson. Real-time correlative scan matching. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4387–4393, 2009.

[10] T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast sparse lidar odometry using self-supervised feature selection on intensity images. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7597–7604, 2022.

[11] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[12] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, pages 834–849, Jul 2014.

[13] H. Wang, C. Wang, C. Chen, and L. Xie. F-loam: Fast lidar odometry and mapping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4390–4396. IEEE, 2021.

[14] H. Ye, Y. Chen, and M. Liu. Tightly Coupled 3D Lidar Inertial Odometry and Mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3144–3150, 2019.

[15] C. L. Gentil, T. Vidal-Calleja, and S. Huang. In2lama: Inertial lidar localisation and mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 6388–.6394, May 2019.

[16] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, Nov 2016.

[17] A. Censi, L. Iocchi, and G. Grisetti. Scan Matching in the Hough Domain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[18] A. Censi and S. Carpin. HSM3D: feature-less global 6DOF scan-matching in the Hough/Radon domain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3899–3906, 2009.

[19] D. Lodi Rizzini. Angular Radon Spectrum for Rotation Estimation. *Pattern Recognition*, 84:182–196, dec 2018.

[20] A. Makadia and K. Daniilidis. Rotation recovery from spherical images without correspondences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(7):1170–1175, 2006.

[21] L. Bernreiter, L. Ott, J. Nieto, R. Siegwart, and C. Cadena. Phaser: A robust and correspondence-free global pointcloud registration. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):855–862, 2021.

[22] H. Yang and L. Carlone. A polynomial-time solution for robust registration with extreme outlier rates. *arXiv preprint arXiv:1903.08588*, 2019.

[23] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[24] J. Shi, H. Yang, and L. Carlone. Robin: a graph-theoretic approach to reject outliers in robust estimation using invariants. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 13820–13827. IEEE, 2021.

[25] J. Straub, T.r Campbell, J.P. How, and J.W. Fisher. Efficient global point cloud alignment using bayesian nonparametric mixtures. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2941–2950, 2017.

[26] W. Clark, M. Ghaffari, and A. Bloch. Nonparametric continuous sensor registration. *Journal of Machine Learning Research*, 22(271):1–50, 2021.

[27] R. Zhang, T.-Y. Lin, C.E. Lin, S.A. Parkison, W. Clark, J.W. Grizzle, R.M. Eustice, and M. Ghaffari. A new framework for registration of semantic point clouds from stereo and rgb-d cameras. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 12214–12221, 2021.

[28] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on Robotics*, 32(6):1309–1332, 2016.

[29] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, et al. The limits and potentials of deep learning for robotics. *Int. Journal of Robotics Research*, 37(4-5):405–420, 2018.

[30] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

[31] Y. Aoki, H. Goforth, R.A. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019.

[32] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[33] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[34] C. Wang, H.K. Galoogahi, C.H. Lin, and S. Lucey. Deep-lk for efficient adaptive object tracking. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 627–634. IEEE, 2018.

[35] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[36] X. Li, J.K. Pontes, and S. Lucey. Pointnetlk revisited. In *Proc. of the IEEE/CVF conf. on computer vision and pattern recognition*, pages 12763–12772, 2021.

[37] A. Kurobe, Y. Sekikawa, K. Ishikawa, and H. Saito. Corsnet: 3d point cloud registration by deep neural network. *IEEE Robotics and Automation Letters (RA-L)*, 5(3):3960–3966, 2020.

[38] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 733–750. Springer, 2020.

[39] Y. Wang and J.M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proc. of the IEEE/CVF Int. Conf. on computer vision*, pages 3523–3532, 2019.

[40] Y. Wang and J.M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. *Advances in neural information processing systems*, 32, 2019.

[41] S. Suwajanakorn, N. Snavely, J.J. Tompson, and M. Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *Advances in neural information processing systems*, 31, 2018.

[42] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart. Challenging data sets for point cloud registration algorithms. *Int. Journal of Robotics Research*, 31(14):1705–1711, 2012.

[43] F. Pomerleau, F. Colas, R. Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015.

[44] Z. Zhang, Y. Dai, and J. Sun. Deep learning based point cloud registration: an overview. *Virtual Reality & Intelligent Hardware*, 2(3):222–246, 2020.

[45] Y. Huang, K.V. Ling, and S. See. Solving quadratic programming problems on graphics processing unit. *ASEAN Engineering Journal*, 1(2):76–86, 2011.

[46] J.V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical programming computation*, 7(3):289–329, 2015.

[47] Edwin Olson. Real-time correlative scan matching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4387–4393, Kobe, Japan, June 2009. IEEE.

[48] C. Roussillon, A. Gonzalez, J. Solà, J.M. Codol, N. Mansard, S. Lacroix, and M. Devy. Rt-slam: a generic and real-time visual slam implementation. In *Int. Conf. on Computer Vision Systems*, pages 31–40. Springer, 2011.

[49] V. Ila, L. Polok, M. Solony, and P. Svoboda. Slam++-a highly efficient and temporally scalable incremental slam framework. *Int. Journal of Robotics Research*, 36(2):210–230, 2017.

[50] M. Abouzahir, R. Latif, M. Ramzi, and M. Sbihi. Opencl and opengl implementation of simultaneous localization and mapping algorithm using high-end gpu. In *ITM Web of Conferences*, volume 46, page 04001. EDP Sciences, 2022.

[51] A. Kumar and R. Chellappa. Disentangling 3d pose in a dendritic cnn for unconstrained 2d face alignment. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 430–439, 2018.

[52] H. Chen, F. Manhardt, N. Navab, and B. Busam. Texpose: Neural texture learning for self-supervised 6d object pose estimation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4841–4852, 2023.

[53] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[54] Andres Milioto and Cyrill Stachniss. Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using cnns. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7094–7100. IEEE, 2019.

[55] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019.

[56] A. Collet and S.S. Srinivasa. Efficient multi-view object recognition and full pose estimation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2050–2055. IEEE, 2010.

[57] A. Collet, M. Martinez, and S.S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *Int. Journal of Robotics Research*, 30(10):1284–1306, 2011.

[58] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3665–3671. IEEE, 2020.

[59] J. Austin, R. Corrales-Fatou, S. Wyetzner, and H. Lipson. Titan: A parallel asynchronous library for multi-agent and soft-body robotics using nvidia cuda. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 7754–7760. IEEE, 2020.

[60] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[61] S.W. Ha and T.D. Han. A scalable work-efficient and depth-optimal parallel scan for the gpgpu environment. *IEEE Trans. on Parallel and Distributed Systems*, 24(12):2324–2333, 2012.

[62] D. Gauthier, P. Freedman, G. Carayannis, and A. Malowany. Interprocess communication for distributed robotics. *IEEE Journal on Robotics and Automation*, 3(6):493–504, 1987.

[63] F. Voigtländer, A. Ramadan, J. Eichinger, C. Lenz, D. Pensky, and A. Knoll. 5g for robotics: Ultra-low latency control of distributed robotic systems. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, pages 69–72. IEEE, 2017.

[64] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 1, pages 476–481. IEEE, 2000.

[65] Y. Chen, X.C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Trans. on Robotics*, 28(1):158–171, 2012.

[66] A. Desai, I. Saha, J. Yang, S. Qadeer, and S.A. Seshia. Drona: a framework for safe distributed mobile robotics. In *Proc. of the 8th Int. Conf. on Cyber-Physical Systems*, pages 239–248, 2017.

[67] S.S. Ali, S. Deka, Z. Ahmad, S. Ahmed, M. Jaswal, and H. Alsulami. Feasibility of drone integration as last mile delivery. *Emerald Emerging Markets Case Studies*, 9(3):1–17, 2019.

[68] S.A Applin. Deliveries by drone: Obstacles and sociability. *The Future of Drone Use: Opportunities and Threats from Ethical and Legal Perspectives*, pages 71–91, 2016.

[69] H. Chen, Z. Hu, and S. Solak. Improved delivery policies for future drone-based delivery systems. *European Journal of Operational Research*, 294(3):1181–1201, 2021.

[70] M.W. Muricho and C.O. Mogaka. Drone technology and performance of retail logistics. *Journal of Sustainable Development of Transport and Logistics*, 7(1):73–81, 2022.

[71] R. Sham, H.X. Chong, E.C. Aw, T.B.T. Thangal, and N. Binti Abdamia. Switching up the delivery game: Understanding switching intention to retail drone delivery services. *Journal of Retailing and Consumer Services*, 75:103478, 2023.

[72] S. Perera, M. Dawande, G. Janakiraman, and V. Mookerjee. Retail deliveries by drones: how will logistics networks change? *Production and Operations Management*, 29(9):2019–2034, 2020.

[73] S. Parrinello and R. De Marco. Digital surveying and 3d modelling structural shape pipelines for instability monitoring in historical buildings: a strategy of versatile mesh models for ruined and endangered heritage. *Acta IMEKO*, 10(1):84–97, 2021.

[74] P. Mahadevan. The military utility of drones. *CSS Analyses in Security Policy*, 78, 2010.

[75] Davide Antonio Cucci, Martin Rehak, and Jan Skaloud. Bundle adjustment with raw inertial observations in uav applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:1–12, 2017.

[76] Arnadi Murtiyoso, Pierre Grussenmeyer, Niclas Börlin, Julien Vandermeerschen, and Tristan Freville. Open source and independent methods for bundle adjustment assessment in close-range uav photogrammetry. *Drones*, 2(1):3, 2018.

[77] C. Gramkow. On averaging rotations. *Journal of Mathematical Imaging and Vision*, 15(1-2):7–16, 2001.

[78] R. Tron, R. Vidal, and A. Terzis. Distributed pose averaging in camera networks via consensus on se (3). In *Second ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pages 1–10. IEEE, 2008.

[79] R. Tron and K. Daniilidis. Statistical pose averaging with non-isotropic and incomplete relative measurements. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 804–819. Springer, 2014.

[80] A. Blair, A.K. Gostar, R. Tennakoon, A. Bab-Hadiashar, and R. Hoseinnezhad. Temporal multiple rotation averaging on a distributed dynamic network. *IEEE Trans. on Signal and Information Processing over Networks*, 2023.

[81] S.H. Lee and J. Civera. Robust single rotation averaging. *arXiv preprint arXiv:2004.00732*, 2020.

[82] F. Dellaert, D.M. Rosen, J. Wu, R. Mahony, and L. Carlone. Shonan rotation averaging: Global optimality by surfing so (p)^ n so (p) n. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 292–308. Springer, 2020.

[83] J. Folkesson and H. Christensen. Graphical slam-a self-correcting map. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 1, pages 383–390. IEEE, 2004.

[84] J. Folkesson and H.I. Christensen. Closing the loop with graphical slam. *IEEE Trans. on Robotics*, 23(4):731–741, 2007.

[85] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[86] D.M. Rosen, L. Carlone, A. S Bandeira, and J.J. Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *Int. Journal of Robotics Research*, 38(2-3):95–125, 2019.

[87] G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Trans. on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[88] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *Int. Journal of Robotics Research*, 31(2):216–235, 2012.

[89] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g 2 o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613. IEEE, 2011.

[90] D.M. Rosen, M. Kaess, and J.J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1262–1269. IEEE, 2012.

[91] L. Carlone, D.M. Rosen, G. Calafiore, J.J. Leonard, and F. Dellaert. Lagrangian duality in 3d slam: Verification techniques and optimal solutions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 125–132. IEEE, 2015.

[92] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert. Initialization techniques for 3d slam: A survey on rotation estimation and its use in pose graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4597–4604. IEEE, 2015.

[93] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[94] F. Arrigoni, B. Rossi, and A. Fusiello. Spectral synchronization of multiple views in se (3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016.

[95] D.M. Rosen, C. DuHadway, and J.J. Leonard. A convex relaxation for approximate global optimization in simultaneous localization and mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 5822–5829. IEEE, 2015.

[96] D.M. Rosen, M. Kaess, and J.J. Leonard. Rise: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Trans. on Robotics*, 30(5):1091–1108, 2014.

[97] M. Cucuringu, Y. Lipman, and A. Singer. Sensor network localization by eigenvector synchronization over the euclidean group. *ACM Trans. on Sensor Networks (TOSN)*, 8(3):1–42, 2012.

[98] Dario Lodi Rizzini and Ernesto Fontana. Rotation Estimation Based on Anisotropic Angular Radon Spectrum. *IEEE Robotics and Automation Letters*, 7(3):7279–7286, 2022.

[99] Ernesto Fontana and Dario Lodi Rizzini. Accurate global point cloud registration using gpu-based parallel angular radon spectrum. *Sensors*, 23(20), 2023.

[100] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2743–2748, 2003.

[101] K.B. Petersen and M.S. Pedersen. The Matrix Cookbook, Nov 2012. Version 20121115.

[102] Y. Liu, C. Wang, Z. Song, and M. Wang. Efficient global point cloud registration by matching rotation invariant features through translation search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 448–463, 2018.

[103] D.F. Crouse, P. Willett, K. Pattipati, and L. Svensson. A look at gaussian mixture reduction algorithms. In *14th International Conference on Information Fusion*, pages 1–8, 2011.

[104] David S Wise, Jeremy D Frens, Yuhong Gu, and Gregory A Alexander. Language support for morton-order matrices. *ACM Sigplan Notices*, 36(7):24–33, 2001.

[105] M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Trans on Visualization and Computer Graphics*, 16(4):599–608, 2010.

[106] R.C. Hoover, A.A. Maciejewski, and R.G. Roberts. Eigendecomposition of images correlated on, and using spectral theory. *Image Processing, IEEE Transactions on*, 18(11):2562–2571, 2009.

[107] X. Bai, X. Yang, L.J. Latecki, W. Liu, and Z. Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(5):861–874, May 2010.

[108] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-Time Loop Closure in 2D LIDAR SLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1271–1278, 2016.

[109] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[110] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 2013.

[111] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, 1965.

[112] K. Malecek and Z Nadenik. On the inductive proof of legendre addition theorem. *Studia Geophysica et Geodaetica*, 45:1–11, 2001.

[113] M.E.H. Ismail. *Classical and Quantum Orthogonal Polynomials in One Variable*. Cambridge University Press, 2009.

[114] E.P. Wigner. Random matrices in physics. *SIAM review*, 9(1):1–23, 1967.

[115] G Aubert. An alternative to wigner d-matrices for rotating real spherical harmonics. *AIP Advances*, 3(6), 2013.

[116] O.H. Ajanki, L. Erdős, and T. Krüger. Universality for general wigner-type matrices. *Probability Theory and Related Fields*, 169:667–727, 2017.

[117] T. Carpentier and A. Einbond. Spherical correlation as a similarity measure for 3-d radiation patterns of musical instruments. *Acta Acustica*, 7:40, 2023.

[118] J. Ivanic and K. Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[119] P.J. Kostelec and D.N. Rockmore. Ffts on the rotation group. *Journal of Fourier analysis and applications*, 14:145–179, 2008.

[120] N. Boumal and P.A. Absil. Rtrmc: A riemannian trust-region method for low-rank matrix completion. *Advances in neural information processing systems*, 24, 2011.

[121] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proc. of the 21st annual conf. on Computer graphics and interactive techniques*, pages 311–318, 1994.

[122] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1–4, 2011.

[123] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2009.

[124] R. Tron, X. Zhou, and K. Daniilidis. A survey on rotation optimization in structure from motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 77–85, 2016.

[125] N. Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[126] N. Boumal, B. Mishra, P.A. Absil, and R. Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[127] C. Criscitiello and N. Boumal. An accelerated first-order method for non-convex optimization on manifolds. *Foundations of Computational Mathematics*, 23(4):1433–1509, 2023.

[128] P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[129] P.A. Absil, R. Mahony, and R. Sepulchre. Optimization on manifolds: Methods and applications. In *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German conf. on Optimization*, pages 125–144. Springer, 2010.

[130] A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[131] P.A. Absil, R. Mahony, and J. Trumpf. An extrinsic look at the riemannian hessian. In *Int. Conf. on geometric science of information*, pages 361–368. Springer, 2013.

[132] T.E. Booth. Power iteration method for the several largest eigenvalues and eigenfunctions. *Nuclear science and engineering*, 154(1):48–62, 2006.

[133] R. Tron, B. Afsari, and R. Vidal. Intrinsic consensus on so (3) with almost-global convergence. In *In proc. of 51st IEEE Conference on Decision and Control (CDC)*, pages 2052–2058. IEEE, 2012.

[134] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar. A distributed optimization framework for localization and formation control: Applications

to vision-based measurements. *IEEE Control Systems Magazine*, 36(4):22–44, 2016.

[135] D.G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London mathematical society*, 16(2):81–121, 1984.

[136] J. Lawrence, J. Bernal, and C. Witzgall. A purely algebraic justification of the kabsch-umeyama algorithm. *Journal of research of the National Institute of Standards and Technology*, 124:1, 2019.

[137] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[138] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1994.

[139] M. Milgram. An addition formula for chebyshev polynomials. *Journal of Physics A-mathematical and General - J PHYS-A-MATH GEN*, 22:323–324, 04 1989.

[140] Legendre polynomials in trigonometry. `https://en.wikipedia.org/wiki/Legendre_polynomials#Legendre_polynomials_in_trigonometry`. Accessed: 2022-10-25.

[141] C. Guarino Lo Bianco et al. *Cinematica dei manipolatori*. Pitagora Editrice, 2004.